НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра автоматики та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК 621.397.63

«До захисту допущено»
Завідувач кафедри
_____ Ролік О. І.
(підпис) (ініціали, прізвище)
“___” _____2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121 Програмна інженері
(код і назва спеціальності)

_____

на тему: Програмні засоби автоматизації проектування сайтів на основі системи управління вмістом

_____

Виконав: студент 6 курсу, групи ІТ-73мп
(шифр групи)

Тучин Владислав Русланович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник проф. Кафедри АУТС Дорошенко А.Ю. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ ас. Кафедри АУТС Шимкович В.М. __ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
“Київський політехнічний інститут
імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Ступінь вищої освіти – другий (магістерський)
(код, назва)

Спеціальність 121 «Програмна інженерія»
(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ролік О. І.
(підпис) (ініціали, прізвище)

“___”_____2018_р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Тучину Владиславу Руслановичу

(прізвище, ім’я, по батькові)

1.**Тема дисертації** Програмні засоби автоматизації проектування сайтів на основі системи управління вмістом

Науковий керівник дисертації__Дорошенко Анатолій Юхимович, проф. Кафедри АУТС_____
затверджені наказом по університету від “_29_”_жовтня_2018_р.№_____

2.Строк подання студентом дисертації “_4_ ” грудня 2018_р.

3. Об`єкт дослідження: системи керування вмістом для розробки веб-сайтів

4. Предмет досліджень: а) Система керування вмістом – 1 ; б) розробка веб-сайтів; в) редагування контенту на сайті та вибор формату виведення; г) вдосконалення інструменту довільних полів.

5.Перелік завдань, які потрібно розробити**:** а) аналіз існуючих рішень; б) формування вимог до проекту; в) розробка діаграм використання програмного розширення; г) розробка структурної схеми; д) розробка бази даних; е) створення розширення з використанням паттерну MVC.
6. Орієнтовний перелік ілюстративного (графічного) матеріалу: структурна схема; схема бази даних для блогу; блок-схема класів; діаграмма використання; діаграмма компонентів; діаграмма послідовності; діаграмма діяльності для створення групи полів; діаграмма діяльності формату виводу.
7. Орієнтовний перелік публікацій

8. Консультанти розділів проекту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

8. Дата видачі завдання "_29_" __жовтня__2018_р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Аналіз існуючих рішень	29.10.2018	
2	Формування вимог до проекту	05.11.2018	
3	Розробка діаграм використання програмного забезпечення	12.11.2018	
4	Розробка структурної схеми	19.11.2018	
5	Розробка бази даних	26.11.2018	
6	Створення розширення з використанням паттерна MVC	26.11.2018	
7	Оформлення текстової та графічної документації	30.11.2018	
8	Представлення до захисту	4.12.2018	

Студент _____ __ Тучин В.Р.

(підпис) (ініціали, прізвище)

Керівник проекту _____ __ Дорошенко А.Ю.
(підпис) (ініціали, прізвище)

УДК 621.397.63

РЕФЕРАТ

Магістерська дисертація: 132 с., 35 рис., 23 табл, 22 дж., 9 додатків.

Метою роботи є розробка програмних засобів для редагування та динамічного виводу контенту в системі керування вмістом WordPress шляхом вдосконалення інструменту – довільні поля.

Об'єкт досліджень: робота з системою керування вмістом, додавання, редагування та видалення контенту на сторінках сайту, робота з форматами виведення контенту, вдосконалення інструменту довільних полів.

У дисертації виконано розробку програмних засобів для роботи з конентом в системі керування вмістом WordPress та аналіз існуючих плагінів, які також допомагають працювати з контентом. Запропоновано новий підхід в роботі з контентом, який реалізований на основі довільних полів у WordPress, що в розробленому плагіні працюють з записами, а не зі сторінками як у конкурентів. Створено розщирення для роботи з контентом на сайті та вибором формату його виведення на сторінках.

Галузь застосування: розробка сайтів на основі системи керуванням вмісту WordPress.

ПЛАГІН, WORDPRESS, КОНТЕНТ, ІНФОРМАЦІЯ, РЕДАГУВАННЯ, ФОРМАТ ВИВОДУ, ПАНЕЛЬ АДМІНІСТРАТОРА, ДОВІЛЬНІ ПОЛЯ, ЗАПИСИ, СТОРІНКИ.

УДК 621.397.63

ABSTRACT

Master's thesis: 112 pp., 35 pp., 23 tables, 22 j., 8 pp.

The purpose of the work is to develop software tools for editing and dynamically outputting content in the WordPress content management system by improving the tool – arbitrary fields.

Object of research: work with the system of management of the contents, adding, editing and deleting content on the pages of the site, work with the format of content output, improvement of the tool of arbitrary fields.

The dissertation is devoted to the development of software tools for working with Constance in the management system of WordPress and analysis of existing plug-ins, which also help to work with content. We propose a new approach to work with content, which is implemented on the basis of arbitrary fields in WordPress, which in the developed plug-ins work with records, and not with the pages as competitors. Creation of an expansion to work with content on the site and the choice of format for its output on the pages.

The scope of application: the development of websites on the basis of the WordPress content management system.

PLAGIN, WORDPRESS, CONTENT, INFORMATION, EDITING, FORMAT VOLUME, PANEL ADMINISTRATOR, CUSTOM FIELDS, POSTS, PAGES.

ЗМІСТ

Перелік умовних позначень, символів, скорочень і термінів

БД – база даних

Довільні поля – параметри які задаються в форматі ключ - значення

Записи – будь-які елементи, які містять в собі дані сайту (контент)

Контент – дані на сторінках сайту

Контроллер – класс, який інтерпретує дії користувача, сповіщаючи модель про необхідність змін

Модель – класс, який надає дані і реагує на команди контролера, змінюючи свій стан

Паттерн – шаблон для проектування программного забезпечення

Сторінки – є контейнером для вмісту, який не залежить від часу

Уявлення – класс, який відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі

ACF – плагін Advanced Custom Fields для редагування контенту

CSS – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки

CMS – система керування вмістом, програмне забезпечення для створення, редагування, організації структури і управління веб-сайтом

JavaScript – скриптова мова програмування

JS – скорочений варіант JavaScript

HTML – стандартизована мова розмітки документів в інтернеті

HTTP – протокол прикладного рівня передачі даних

MVC – схема поділу даних додатка

MySQL – вільна реляційна система управління базами даних

PHP – скриптова мова загального призначення, інтенсивно застосовується для розробки веб-додатків

UML – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, системного проектування та відображення організаційних структур

WordPress – система управління вмістом сайту з відкритим вихідним кодом

ВСТУП

Серед розробників сайтів, великою популярністю користується система управління сайтів або движок WordPress (Вордпресс). WordPress – система управління сайтом з відкритим вихідним кодом, реалізована на мові програмування php і використовує стандартну базу даних MySQL. Незважаючи, на те, що дана платформа, створювалася в першу чергу для блогів, сучасні можливості WordPress дозволяють створювати сайти будь-якої складності, від корпоративних сайтів до інтернет-магазинів або порталів. Згодом система набрала популярність і під неї стали створювати додаткові плагіни і теми, адаптувавши архітектуру WordPress під більш складні проекти. WordPress заслужив високу оцінку в розробників, клієнтів, пошукових системах завдяки ряду переваг.

Простота в експлуатації і використанні, завдяки грамотно розробленій адміністративній панелі. Це дає можливість вносити зміни і додавати матеріали людям без додаткових знань програмування. Також величезна кількість безкоштовних тем з різноманітним дизайном, шаблонів і більше 10 тис. модулів дозволяють вибрати і, в разі необхідності, вдосконалити, змінити, поліпшити розробку або роботу будь-якого веб-ресурсу.

WordPress має дуже зручне навігаційне меню, яке можна легко адаптувати під основні завдання сайту, тим самим покращивши його юзабіліті і сприйняття користувачами. Система постійно оновлюється, і вдосконалюється, а разом з нею і веб-сайт власника.

Сайт на WordPress[1] відмінно індексується пошуковими системами, що дає можливість покращувати рейтингові показники сайту, при пошуковому просуванні ресурсу в ТОП 10. Дана платформа дає можливість внесення змін вибраного шаблону і створення унікального сайту по дизайну або структурі. За рахунок цього на ринку велика кількість розробників володіє WordPress, відповідно великий вибір підрядника на розробку і нижча вартість робіт.

Таким чином, у разі прийняття рішення по розробці власного сайту-візитки,

корпоративного сайту, блогу або простого інтернет-магазину при виборі движка варто в першу чергу звернути увагу на WordPress. Використовуючи безкоштовний шаблон WordPress, розробка сайту, що продає забере не багато часу і фінансових витрат, а в результаті клієнт отримає відмінний інструмент для продажу власних товарів або послуг в мережі інтернет.

Розширення для редагування і динамічного виведення інформації в WordPress називається плагіном. Аж до версії WordPress 1.2 можливість зміни його функціоналу «під свої потреби» або розширення можливостей досягалися шляхом редагування вихідного коду ядра платформи WordPress. Але це створювало різні незручності (наприклад, при оновленні версій), і від такої практики незабаром відмовилися. Розробники впровадили досить зручну, зрозумілу і легку у використанні програмістами систему розширення функціоналу за допомогою «плагінів». Основна ідея використання нової системи розширення можливостей полягала в тому, щоб зберігати ядро цілісним і незмінним і в той же час дати php-програмістам можливість змінювати його поведінку за допомогою спеціальних легко підключающихся (і відключающихся) скриптів-плагінів.

Плагін WordPress – це програма або набір функцій, написаних на php, що додають певний набір можливостей або сервісів до веб-сайту на WordPress, які легко об'єднуються з системою управління та функціоналом WordPress за допомогою Plugin Application Program Interface (API).

Якщо ви хочете додати або змінити будь-яку функціональність Wordpress, перше, що вам потрібно зробити, це пошукати вже готові рішення в офіційному сховищі плагінів від творців CMS – може бути, хто-небудь вже створив плагін, який вирішує завдання користувача. Якщо ж знайти необхідне рішення не вдалося, то його можна створити самостійно.

Метою даної роботи є розробка плагіна для CMS WordPress, який дозволить власнику сайту за допомогою адміністративної панелі редагувати вміст на необхідних сторінках веб-сайту і виводити його в потрібному вигляді. Завдяки даному розширенню

після здачі проекту розробника сайту, власник, який не розбирається в програмуванні зможе змінити динамічне відображення потрібних блоків з контентом і вивести його на будь-які сторінки.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

У WordPress[2] за замовчуванням є можливість для кожного поста (будь-якого типу, будь то запис post, сторінка page або який-небудь довільний тип постів) задавати необмежену кількість параметрів. На цей випадок є навіть окрема таблиця в базі даних wp_postmeta.

Ці параметри називаються довільними полями або метаданими поста. Метадані складаються з пар ключ => значення, наприклад color => червоний, size => великий.

Засобами CMS довільні поля створюються за допомогою функцій або через панель адміністратора. Перший спосіб підійде тільки програмістам, але прицьому трудомісткий, адже для кожного поля описується конкретний метод. Другий спосіб підійде навіть власнику сайту, який не володіє навичками веб-програмування, але має багато часу, адже для кожної сторінки або окремого запису потрібно заново створювати поля, відсутня можливість спочатку задати набір полів і потім виводити їх в потрібному місці.

Розроблений плагін в першу чергу вирішує проблему відсутності зручності в роботі з довільними полями. Досягається це шляхом створення групи полів. На сторінці налаштувань спочатку створюється група полів, потім в ній створюються конкретні поля, після чого можна в налаштуваннях вибрати на яких сторінках або записах їх виводити. Все це працює через панель адміністратора і є принципово новим підходом, який відрізняється від того, що пропонує WordPress за замовчуванням.

Також можливості динамічно міняти відображення даних за замовчуванням немає. Існуючі плагіни в офіційному сховищі WordPress також відсутні. Розроблене розширення пропонує зручний спосіб динамічного виведення інформації на основі готових шаблонів. Після виведення групи полів на сторінці або конкретному записі в

них же можна вибрати дизайн для виведення будь-якого поля. Все це робиться за допомогою призначеного для користувача інтерфейсу. Спочатку коли сторінка верстається, вона має свою структуру. У неї програміст додає вивід довільних полів. Представлений плагін змінює тільки верстку в залежності від обраного шаблону, залишаючи методи з підключенням довільних полів на своєму місці. В результаті можна продовжувати редагувати інформацію з панелі адміністратора, але при цьому її відображення на сторінках сайту вже буде таким, яке вибрав користувач. Порівняння з існуючими розширеннями описано далі в магістерській дисертації і наочно показує відмінності і новизну представленого рішення.

Для проектування плагіна був обраний принципово нових підхід на основі патерну MVC, що відрізняється від класичного підходу на основі процедурного програмування шляхом написання методів для дій, всередині яких знаходиться логіка плагіна. У представленому розширенні для редагування і динамічного виведення інформації вся логіка виведена в моделі, реакція плагіна на дії користувача в контролери, а призначений для користувача інтерфейс та шаблони для виведення інформації винесені в уявлення. Все це пов'язано в одному керуючому файлі. Також розроблені компоненти уявлень для окремих модулів розширення.

Такий підхід допоміг спростити розробку і структурувати код, в подальшому розширити функіцонал плагіна і прискорити завантаження окремих елементів розширення в процесі роботи з ним.

1.1 Актуальність

На даний момент в числі популярних плагінів для спрощеного створення довільних полів лідирує Advanced Custom Fields. Розширення розроблено незалежними розробниками, які допомагають програмістам значно простіше працювати з довільними полями, але можливості плагіна обмежуються тільки редагуванням даних на сторінках.

Головною перевагою розширення для редагування і динамічного виведення інформації є зміна зовнішнього вигляду даних в будь-яких блоках сторінки з панелі

адміністратора. Йдеться про те, що власник ресурсу зможе змінювати верстку конкретних блоків не заглядаючи в код і не звертаючись за допомогою до програмістів. Наприклад, на одній зі сторінок сайту присутня "галерея", де після завершення розробки виводиться 4 фотографії по 2 в ряд. Якщо їх кількість не зміниться, то це буде зручний і привабливий формат для перегляду, але в разі регулярного додавання нових фотографій все зміниться. Як мінімум, користувачам для галереї з такою структурою доведеться довго гортати сторінку сайту і скоріше за все їм це швидко набридне і до інформації, яка розташована в наступному блоці за розділом «галерея», вони не доберуться.

У такому випадку власник сайту повинен змінити верстку і тим самим відобразити фотографії в більш зручному форматі, але, як правило, він сам не знає, як це зробити. Можна звернутися до програмістів, але якщо це робити постійно, то легко втратити багато грошей на рішення таких простих задач. Розширення для редагування і динамічного виведення інформації вирішує проблему, оскільки використовується кілька вбудованих шаблонів. Вони спроектовані на основі розробленої 12-колоночної сітки. Створювалося таке рішення з нуля, щоб в процесі використання плагіна на вже готових сайтах не було конфлікту в стилях CSS. Зараз велика частина веб-сайтів робиться з застосуванням фреймворків, які користуються своїми сітками і багато плагінів від недосвідчених розробників в результаті з ними конфліктують.

Розширення пропонує кілька способів відображення контенту для яких спеціально створювалися шаблони. У процесі створення через плагін групи довільних полів в неї додається поле select з варіантами шаблонів для конкретного блоку даних, які потім власник сайту зможе вибрати через панель адміністратора. Якщо повернутися до прикладу з розділом "Галерея", то спочатку картинки будуть розміщені по вбудованому, в розроблений для магістерської дисертації плагін, шаблону, наприклад, під назвою "в 2 колонки". Виходить по 2 фотографії в ряд. Згодом власник сайту вирішує, що йому потрібно більше фото і ось їх уже стало 12 замість тих 4 що були відразу. За допомогою плагіна, він в панелі адміністратора просто вибирає шаблон, наприклад, під назвою "в 3 колонки" (всі назви будуть продумані для того, щоб вони були інтуїтивно зрозумілі

користувачеві). Тепер фотографії розміщені по 3 штуки в ряд, що значно зручніше для перегляду, частково змінюється структуру сайту, адже тепер блок "галерея" по висоті займає менше місця і користувач однозначно добереться до наступних блоків на сайті.

Плагін стане і помічником в тому випадку якщо власник сайту просто вирішить освіжити свій ресурс і трохи поміняти його структуру за допомогою вбудованних шаблонів. Він зможе це робити в будь-який час, адже всі дії здійснюються через адміністративну панель з інтуітивно зрозумілим, призначеним для користувача інтерфейсом.

На жаль, раніше на офіційному сховищі плагінів WordPress не з'являлося подібного інструменту для створення довільних полів і подальшої динамічної зміни виведення інформації. Розробка розширення для магістерської дисертації спробує вирішити дану проблему, тим, що спростить процес розробки сайтів під WordPress. За допомогою плагіна програміст зможе створити групу довільних полів і вивести в них дані для подальшого редагування з боку користувача. Також розширення допоможе власникам веб-сайтів, які в подальшому захочуть міняти саме структуру даних на сторінках, робити це самостійно без залучення фахівців.

Ідея створення розширення для редагування і динамічного виведення інформації була викликана тим, що в WordPress немає необхідних функцій для цього і навіть не вдалося знайти відповідні плагіни в офіційному сховищі, щоб вирішити поставлені задачі. Розробляючи комерційні сайти в обов'язковому порядку потрібно давати можливість власникам змінювати інформацію. При цьому після розробки більше десятка проектів, був зроблений висновок, що динамічний вивід інформації за допомогою панелі адміністратора є дуже важливою функцією. Аналогів представленого в магістерській дисертації на даний момент немає, але існують плагіни, які виконують певні функції з переліку поставлених завдань в розробці. Першим є плагін Advanced Custom Fields (ACF).

Розробники ACF не придумали ніякого нового функціоналу, тому що додавання довільних полів є стандартною функцією WordPress. Вони створили плагін, який

дозволяє додавати поодинокі довільні поля зі зручним інтерфейсом і можливістю їх налаштування. У вільному доступі поширюється безкоштовна версія з обмеженим функціоналом. Також є платна версія "PRO", яка збільшує кількість різновидів довільних полів. Основна перевага ACF полягає в тому, що плагін дозволяє робити поля різних типів, наприклад, текстових, для зображень або списків, а також є спеціальні готові поля під інтерактивні карти, слайдери та інше. До можливостей ACF належать:

- вставка довільних полів;
- вибір місця куди вставити поле;
- налаштування області редагування запису.

Розроблене розширення для магістерської дисертації також вирішує ці завдання, але при цьому володіє більш простим призначеним для користувача інтерфейсом і зрозумілими кожному налаштуваннями. Також відмінністю є можливість групувати поля. Програміст зможе заздалегідь створити необхідні поля, потім об'єднати їх в групу і підключити її в панелі адміністратора на потрібній сторінці. Завдяки цьому користуватися довільними полями власнику веб-сайту буде значно зручніше. Крім цього розширення пропонує динамічний вивід інформації на сторінках, чого в ACF немає.

Другим плагіном для більш вільної роботи з сайтом надалі є Toolset Types. Він дозволяє власнику створювати "довільні типи записів" з особливими налаштуваннями. Якщо за умовчанням в WordPress запис містить тільки заголовок, основну частину і мініатюру, то за допомогою даного плагіна його можливості можна розширити. До можливостей Toolset Types можна віднести:

- створення призначених для користувача типів записів;
- зберігання в записах будь-якого типу інформації;
- використання існуючої бази даних без зміни її структури;
- легко адаптувати записи під SEO.

Незважаючи на те, що працюючи з плагіном можна легко створити унікальний тип запису зі своїми довільними полями, які в подальшому власник веб-сайту зможе легко

редагувати, у нього є недоліки. По-перше, працює розширення тільки з записами і не дає можливість змінити дані на статичних сторінках, що важливо для всіх сайтів, крім односторінкових візиток. По-друге, досить складні налаштування конкретного типу в процесі створення, що власнику веб-сайту може бути не зрозуміло. По-третє, немає ніяких інструментів для роботи з динамічним виведенням інформації в записі або сторінці.

Разроблене розширення для магістерської дисертації не виділяє призначені для користувача типи даних, але це можна зробити завдяки додаванню до конкретних записів і сторінок групи полів. При бажанні можна об'єднати записи в певну рубрику, додати групу полів саме для рубрики і по суті це буде аналогічно роботі Toolset Types, тільки в рази простіше через відсутність складних налаштувань. Також розроблене розширення пропонує динамічний вивід інформації, чого немає в Toolset Types.

Єдиним схожим аналогом з представленим розширенням може бути тільки плагін-конструктор сторінок. З ним можна відмовитися від довільних полів і зібрати сторінку за допомогою вбудованних інструментів. Сучасні рішення такого типу навіть дають можливість зробити унікальний дизайн для спроектованих сторінок. Якщо власнику веб-сайту знадобиться змінити інформацію, то він зможе це зробити через панель адміністратора в розділі самого конструктора. Проте динамічного виведення інформації в ньому також немає. Якщо потрібно поміняти структуру конкретної сторінки, то доведеться вибрати потрібний блок в конструкторі і заново перебудувати його, що займає час і причому в цей момент роботу сайту доведеться зупинити. При використанні розширення описаного в магістерській дисертації таких проблем немає. До групи довільних полів на сторінці додається вибірка шаблонів для конкретного блоку інформації. Поміняти зовнішній вигляд можна вибравши будь-який з доступних видів в один клік, після чого натиснути на сторінці або записі "оновити" і зміни вступлять в силу без зупинки роботи веб-сайту. Варто відзначити, що плагіни конструктори пропонують обмежену кількість інструментів для конструювання блоків на сторінці, причому без можливості розширити функціонал для користувача. У роботі з представленим плагіном

були вивчені інструменти проектування шаблонів в конструкторах, потім вдосконалені і реалізовані, щоб не поступатися даним розширенням. Також розширення для редагування і динамічного виведення інформації передбачає тривалий розвиток, в процесі якого будут з'являтися нові можливості.

1.2 Практична цінність

Створення унікального сайту починається з розробки дизайну, який потім передається програмісту для верстки. Надалі, щоб його можна було використовувати на CMS WordPress необхідно привести файли до загального вигляду, які називаються темою. На офіційному сайті WordPress є чітка інструкція для створення теми і інформація про те, яку вона повинна мати структуру. Далі тема завантажується через панель адміністратора і активується.

Активована тема надає можливість тільки додавати нові записи та сторінки, але вони все одно будуть статичними і змінити вміст або зовнішній вигляд такої сторінки власник веб-сайту не зможе. У цьому допоможе одна з функцій росширення для редагування і динамічного виведення інформації з магістерської дисертації. Перш за все це розширення використовується для створення групи довільних полів для кожної сторінки.

Довільні поля будуть заповнюватися розробником сайту тією інформацією, яку в подальшому власник ресурсу захоче редагувати самостійно. Це може бути текст, картинки, відео, слайдери і т.п. У розробленому плагіні створення групи полів і наповнення її конкретними полями відбувається через панель адміністратора, що дуже зручно. Щоб поле запрацювало, необхідно додати спеціальний код в верстку. Після здачі веб-сайту в експлуатацію замовника йому не доведеться переглядати код сторінок для того, щоб змінити інформацію, досить перейти в потрібний розділ в панелі адміністратора.

Росширення вирішує в першу чергу проблему редагування даних на сторінках сайту, яка виникає у тих хто не розбирається в програмуванні. Оскільки після розробки

веб-сайт може працювати без зміни зовнішнього вигляду або структури протягом 3-4 років, то редагування даних є просто необхідним заходом.

Науковому керівнику в даній магістерській дисертації належить постановка задачі та основні рекомендації щодо її виконання. Також поради щодо реалізації плагіна і складання документації.

Магістрантом була обрана конкретна тема, тобто створення розширення для виведення і динамічного редагування інформації в CMS WordPress. Також вибір паттерна MVC для проектування плагіна і його впровадження для вирішення поставлених завдань. Разом з цим реалізація магістерської дисертації, тестування на конкретному прикладі (web-сайті) і складання документації.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОЕКТУ

За замовчуванням WordPress пропонує тільки редагування «записів» (стандартний запис, який дозволяє зробити на сайті, наприклад, новини, нотатки, відгуки і подібні види інформації). Причому змінити можна тільки заголовок запису і його зміст.

WordPress дає можливість створювати нові сторінки всередині теми, але вони не є чимось унікальним і їх зовнішній вигляд повністю повторює стиль встановленої теми. Тема Wordpress – це набір файлів, які відповідають за зовнішній вигляд вашого блогу в браузері, те, як він буде виглядати і буде оформлений. Ці файли дають браузеру інформацію про те, яким чином, в яких місцях потрібно виводити інформацію і як її відображати.

Якщо людина хоче отримати унікальний веб-сайт, то перш за все вона звертається до дизайнера, який "малює" нові сторінки для майбутнього ресурсу. Після цього макети потрапляють в руки веб-програміста, який може зробити для WordPress унікальні сторінки, які називаються шаблонами. Поняття шаблон вужче, ніж тема – це ті файли всередині теми, які формують вивід інформації потрібним чином. Наприклад, шаблон для виведення записів блогу, шаблон для виведення сторінок, шаблон для виведення рубрик і т.д. Це узагальнений варіант, оскільки зараз можна створювати унікальні

шаблони під кожну сторінку. Прикладом для комерційного сайту будуть сторінки: наша команда, контактні дані, проекти, послуги і т.д. Шаблон, по суті, це просто файл з розширенням php, який містить в собі html-код і php-функції.

Незважаючи на те, що сучасні сайти на WordPress можна робити з унікальним зовнішнім виглядом цього все одно мало для майбутнього власника. Після отримання готової роботи, він захоче міняти текст, картинки, відео та іншу інформацію на сторінці самостійно. Саме для цих цілей і буде використовуватися плагін в даній магістерській дисертації. В цілому можливості нового розширення наступні:

- додавання довільних полів. З їх допомогою дані зберігаються парою, ключ – назва довільного поля і значення – вміст поля, яке виводиться на сторінці або всередині запису;
- формування групи довільних полів для того, щоб їх можна було зручно використовувати для конкретної категорії записів;
- налаштування групи довільних полів під себе;
- іменування довільних полів;
- редагування інформації всередині довільного поля;
- формат відображення групи полів для конкретного блоку даних.
- автоматичне створення шаблону для певного формату виведення.

Як правило, плагіни для WordPress розробляються з використанням процедурного програмування. Такий підхід не зручний якщо мова йде про досить велике за функціоналом розширення. В даному випадку мова йде про багатофункціональне розширення з декількома самостійними можливостями, тому було прийнято рішення спроектувати його за допомогою паттерна MVC.

Використовуючи шаблон проектування вдалося розділити програмний код на логічні блоки, що допоможе без проблем розширити функціонал надалі. Логіка плагіна

описана в моделях. Всі можливості в розширенні мають свою модель, яка працює, як окремий модуль. Всі дії користувача в роботі з плагіном описані в окремих контролерах. До них можна віднести установку конкретних налаштувань для плагіна, вибір шаблону безпосередньо з панелі адміністратора і інші дії, які пов'язані з відправкою користувачем звернень до сервера.

Вся візуальна складова роботи плагіна винесена в уявлення. Йдеться про налаштування плагіна з панелі адміністратора, інструменти призначені для користувацького інтерфейсу для вибору шаблонів і подальшого динамічного виведення інформації.

Завдяки паттерну MVC вдалося зв'язати всі компоненти розширення максимально зручним способом. В папці models розмістилася логіка для налаштувань розширення, підключення до бази даних, виведення управління плагіном в панель адміністратора і елементів управління для користувача, які виводяться на сторінках, де підключені довільні поля.

Контролер містить опис роботи плагіна при конкретних діях. Кожен плагін в WordPress має як мінімум 3 базових стани. Активація – стан, коли користувач приводить розширення в роботу. Деактивація – відключення плагіна від сайту. Видалення – очищення всіх файлів плагіна для звільнення місця всередині CMS. У процесі розробки були розроблені контролери для всіх цих дій, а також допоміжні, які допомогли значно простіше реалізувати вибір конкретного шаблону з панелі адміністратора та багато іншого.

В процесі розробки було прийнято рішення використовувати уявлення для створення шаблонів. Таким чином папка views містить базові шаблони розширення з готовою версткою для динамічного відображення контенту. Також в неї увійшла візуальна складова плагіна в адміністративній панелі – сторінка налаштувань. Після активації будь-якого плагіна його необхідно налаштувати під свої потреби. У даній ситуації сторінка налаштувань містить вибір базових конфігурацій, створення груп полів, створення довільних полів і підключення їх на конкретній сторінці або записах.

Нововведенням в процесі проектування стало використання так званих компонентів уявлення. Вони дозволили виділити логіку окремих компонентів плагіна в окремі файли для більш зручної роботи і максимально простого розширення в майбутньому. В плагіні вони використовуються для формування групи полів і виведення елементів управління для записів і сторінок з вибором шаблону, який в підсумку динамічно виводить інформацію.

Створення компонентів уявлення допомогло розділити деякі елементи функціонала на модулі. Їх вдалося вивести в такому вигляді всередині ключових уявлень. Такий підхід допоможе в подальшому розширювати окремі ділянки коду, прискорює роботу плагіна за рахунок того, що компоненти уявлення сервер знаходить швидше за все. Структура DOM дерева (структура веб-сторінки) така, що читає код зверху вниз і відповідно в такій послідовності йде підключення методів. Замість того, щоб розміщувати окремі модулі програмного коду у вигляді методів, використовуються компоненти уявлення, які на уявленні обробляються браузером, а потім вже все інше.

3 СЦЕНРАІЇ ВИКОРИСТАННЯ СИСТЕМИ

Для того, щоб краще зрозуміти роботу плагіна було разраблено кілька UML діаграм[3], які наочно показують функціонал розроблюваного розширення. Перш за все інформація описана у вигляді діаграм допомагає розібратися в тому, як користувач буде взаємодіяти з плагіном вже на готовому сайті.

3.1 Діаграма використання (Use case)

Діаграма використання необхідна, щоб відповісти на головне питання моделювання: що робить розщирення в зовнішньому світі. На діаграмі використання застосовується 2 типу основних сутностей: варіанти використання (активація, створення группи полів, вибір формату виводу і т.д.) і дійові особи (користувач), між якими встановлюються зв'язки типу асоціація.

Асоціація між дійовою особою і варіантом використання демонструє, що дійова особа якимось чином взаємодіє (надає подібні дані або отримує результат) з варіантом використання. Тобто асоціація показує, що дійова особа в будь-якому випадку бере участь у виконанні кожного із сценаріїв, які

описані варіантом використання. Асоціацію можна вважати найбільш важливим і практично завжди обов'язковим відношенням на діаграмі використання. Якщо на діаграмі немає асоціації між дійовою особою і варіантом використання, то це означає, що система, в нашому випадку плагін, не взаємодіє з зовнішнім світом.

Рисунок 3.1 – Діаграма використання

Діаграма використання показує, як дійова особа, в нашому випадку користувач сайту, який зайшов в панель адміністратора, взаємодіє з плагіном. Спочатку у нього є 3 дії: активація, деактивація і видалення плагіна. Щоб перейти до роботи з плагіном йому потрібно для початку його активувати, після цього, він переходить до налаштувань.

Плагін працює таким чином, що перед додаванням контенту необхідно створити групу полів. У неї заносяться поля, в яких буде зберігатися майбутній контент і потім виводитися на конкретній сторінці. Користувач може вибрати потрібні типи полів, додати яку завгодно кількість в групу і створити її. Після цього йому потрібно вибрати цільову сторінку, тобто ту, де поля будуть відображатися в панелі адміністоратора.

Наступна дія це додавання контенту, користуючись панеллю адміністратора користувач переходить на цільову сторінку і бачить, що плагін вже вивів для неї групу полів, але вони порожні. Тепер необхідно їх заповнити.

Після наповнення полів інформацією у користувача з'являється можливість відредагувати дані при необходіомсті і вибрати формат виводу. Формат виведення – це по суті варіант шаблону, за яким і буде виводиться контент вже на сторінці сайту в браузері.

Останньою дією є публікація. На діаграмі використання відображені всі можливі дії користувача, щоб наочно показати, як він взаємодіє з плагіном в системі керування вмістом WordPress.

3.2 Діаграма діяльності

Діаграма діяльності – спосіб опису поведінки на основі вказівки потоків управління і потоків даних. Візуально такий спосіб опису поведінки нагадує блок-схему алгоритму. Проте за рахунок модернізованих позначень, узгоджених з об'єктно-орієнтованим підходом, а головне, за рахунок нової семантичної складової, діаграма діяльності UML є потужним засобом для опису поведінки системи.

На діаграмі діяльності застосовується один основний тип сутностей – дія (створення группи полів), і один тип відносин – переходи (передачі управління і даних). Часто використовуються такі конструкції як розвилки, злиття, з'єднання, розгалуження, які схожі на сутності, але такими насправді не є, а є графічним способом зображення деяких окремих випадків багатомісних відносин.

У даній ситуації була розроблена діаграма діяльності для демонстрації створення группи полів та заповнення їх контентом з подальшим збереженням на цільовій сторінці після активації плагіну, оскільки це одна з можливостей роботи розробки у магістерській дисертації. Також розроблена друга діаграма діяльності для відображення процессу вибору формату виводу контенту в группі полів.

Рисунок 3.2 – Діаграма діяльності для створенная группи полів

Діаграма використання показує процес створення групи полів. Саме з цього починається робота з плагіном і це його основна функція. Розширення для WordPress спрощує роботу зі стандартними довільними полями, щоб в подальшому було простіше взаємодіяти з контентом.

Після активації плагіна користувач переходить до налаштувань і створює групу полів. Це перший процес на діаграмі, який переходить до додавання конкретних полів в групу і вибору цільової сторінки. Все це відбувається на одній сторінці налаштувань – паралельно. Тобто в одному блоці додаються поля, в іншому тут же можна вказати цільову сторінку.

Коли поля створені і обрана сторінка, користувачеві залишається тільки зберегти групу полів. Після цього потрібні поля для роботи з коентом з'являються вже на цільовій сторінці і для подальшої роботи потрібно перейти до неї, користуючись панеллю адміністратора.

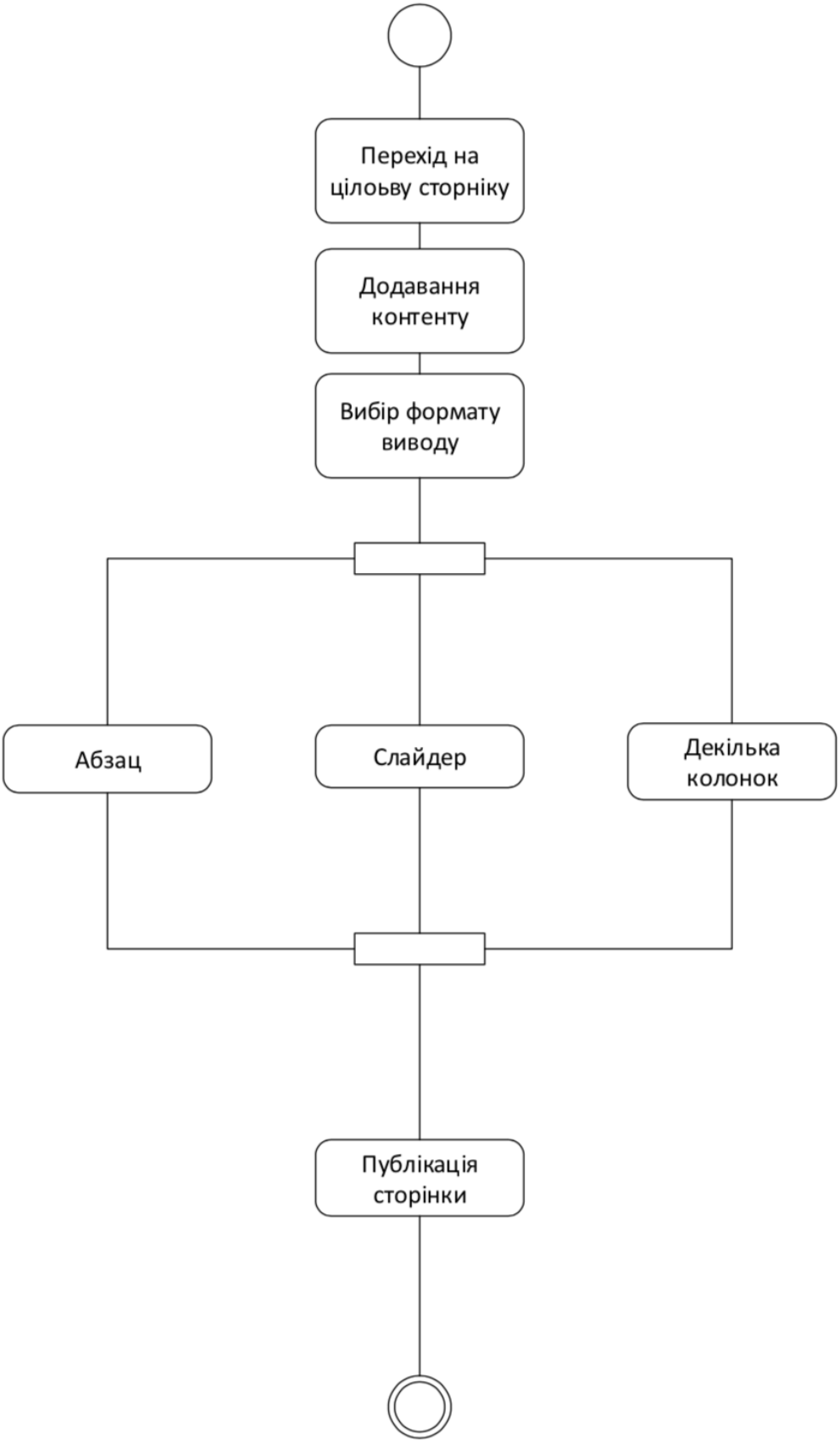


Рисунок 3.3 – Діаграма діяльності для вибору формату виводу

Наступна діаграма використання показує процес вибору формату виведення контенту, який доданий до групи полів.

Щоб почати працювати з групою полів, яка була створена раніше, за допомогою плагіна потрібно для початку перейти на цільову сторінку, до якої прив'язана група полів.

На сторінці в панелі адміністоратора користувач може наповнити довільні поля в створеній ним групі контентом. З цільової сторінки, також можна редагувати інформацію в полях, якщо це буде потрібно.

Коли поля в групі заповнені необхідно вибрати формат виводу, щоб визначити як контент буде виглдеть на сторінці сайту в браузері. У діаграмі представлено 3 формату: абзац, кілька колонок і слайдер. Для початку в плагіні реалізовані тільки базові формати. Абзац є фрагментом тексту, який розташовується по всій ширині сторінки. Кілька колонок – це формат, коли інформація розподіляється по колонках і вони йдуть в один ряд, можна буде вибрати кілька варіантів, які відповідають сітці grid, оскільки саме вона використовується для верстки веб-сторінок. Слайдер представляє собою зручний формат виведення картинок, до яких можна додавати і текстовий опис, подібний функціонал на сайтах вирішується використанням сторонніх бібліотек, але в плагіні це є базовою можливістю.

Останнім кроком після вибору формату виведення є публікація сторінки, в адміністративній панелі WordPress – це по суті означає, що вміст сторінки зберігається і відображається вже в браузері.

Надалі кількість форматів виводу буде збільшено, щоб плагін вирішував якомога більше рутинних завдань без втручання розробника.

3.3 Діаграма послідовності

Діаграма послідовності – це спосіб опису поведінки системи на основі вказівки послідовності переданих повідомлень.

На ділі діаграма послідовності являє собою запис протоколу конкретного сеансу роботи системи. В об'єктно-орієнтованному програмуванні найістотнішим під час виконання є пересилання повідомлень між взаємодіючими об'єктами. Саме послідовність посилок повідомлень відображається на даній діаграмі, звідси і назва.

На діаграмі послідовності використовується один основний тип сутностей – екземпляри взаємодіючих класифікаторів (Користувач,
Сайт, Панель адміністратора), і один тип відношень зв'язків, за якими відбувається обмін повідомленнями. Існує кілька видів повідомлень, які в графічній нотації розрізняються видами стрілок, відповідного відношення.

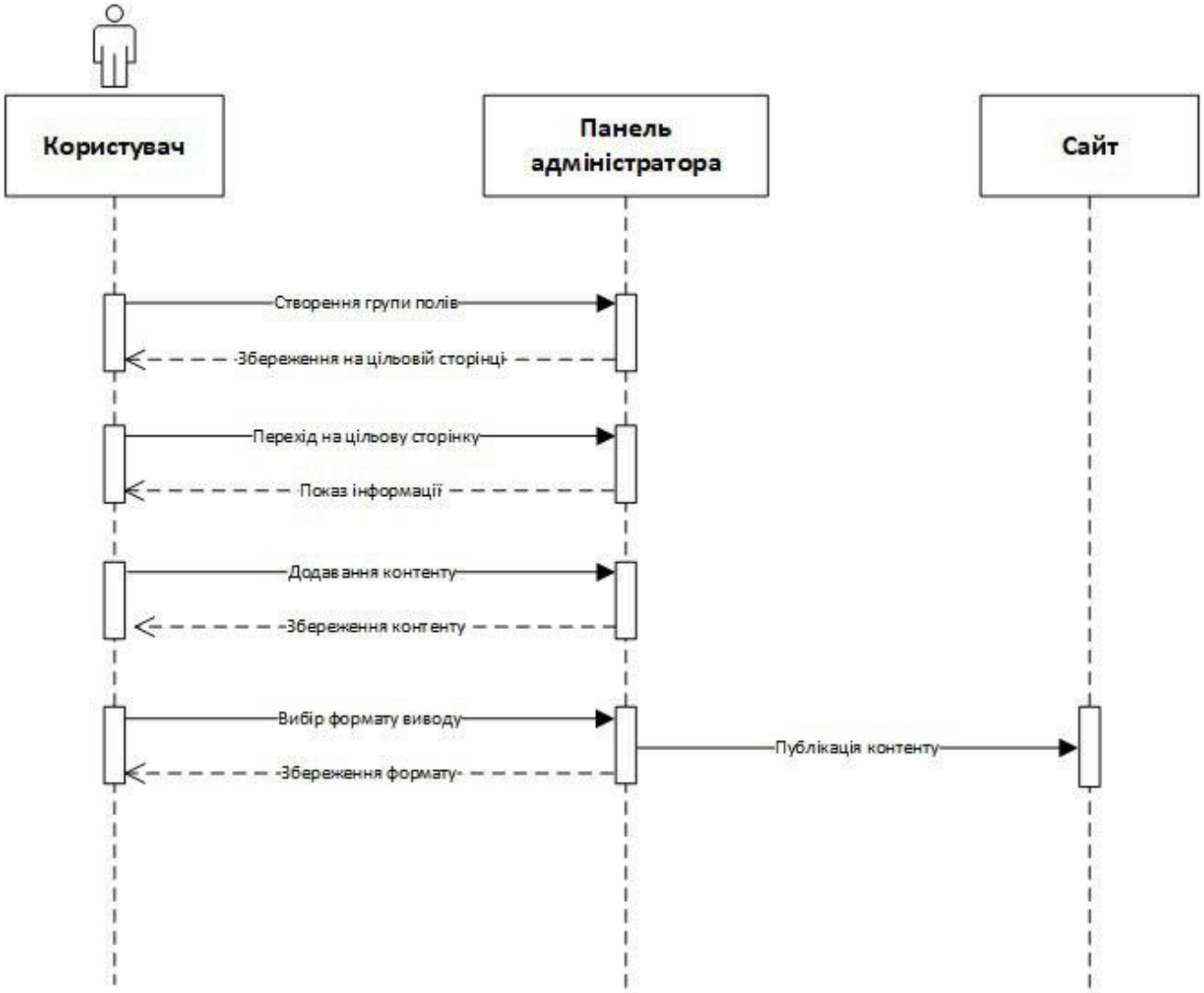


Рисунок 3.4 – Діаграма послідовності

Діаграма послідовності наочно показує, як відбувається обмін повідомленнями при виклику різних функцій плагіна.

Користувач звертається до панелі адміністратора, щоб створити групу полів і у відповідь отримує збереження групи на цільовій сторінці. Також він може зробити запит переходу на потрібну йому сторінку і тоді панель адміністратора відповість йому показом данних, який міститься на сторінці інформації.

Крім цього користувач робить запит до панелі адміністратора, щоб додавати контент в поля і вибирати формат виведення. У відповідь отримує збереження і тих і інших даних.

При цьому панель адміністратора після збереження формату виведення і контенту в групі полів відправляє запит самому сайту з публікацією контенту.

Таким чином, за допомогою діаграми послідовності можна відстежити які запити може відправляти користувач до панелі адміністратора, користуючись плагіном і які з них йдуть вже від WordPress до самого веб-сайту в браузері.

3.4 Діаграма компонентів

Діаграма компонентів – показує взаємозв'язки між модулями (логічними або фізичними), з яких складається змодельована розробка.

Основний тип сутностей на діаграмі – це самі компоненти (DBMS – система управління базами даних), а також інтерфейси, за допомогою яких вказується взаємозв'язок між компонентами. На діаграмі компонентів застосовуються такі відносини:

- реалізації між компонентами і інтерфейсами (компонент реалізує інтерфейс);
- залежність між компонентами і інтерфейсами (компонент використовує інтерфейс).

Рисунок 3.5 – Діаграма компонентів

Діаграма компонентів показує як модулі, з яких складається розробка пов'язані між собою. Всі дані плагіна зберігаються в окремій базі даних і для їх управління є вбудована система управління (DBMS) під назвою phpmyadmin. Вона є частиною WordPress і спрощує роботу з базами даних, а саме створення баз, таблиць, записів в таблицях і написання запитів до бази на мові SQL.

Система управління базами даних безпосередньо пов'язана з сервером, який обробляє запити, після чого відправляє результат в браузер. Відповідно, на діаграмі сервер пов'язаний з системою упрваленія базами даних.

Останньою ланкою в діаграмі є веб-браузер, який відображає отриману від бази даних інформацію на сторінках для того, щоб її побачив користувачів.

Всі дії, які виконуються в плагіні так чи інакше відправляються в базу даних, потім обробляються сервером і виводяться на сторінках готового сайту у вигляді контенту.

4 СТРУКТУРНА СХЕМА СИСТЕМИ

На самому початку розробки була створена структурна схема для того, щоб чітко розуміти з чим доведеться працювати і розбити створення плагіна на конкретні блоки.

Структурна схема – це сукупність елементарних ланок об'єкта і зв'язків між ними, один з видів графічної моделі. Під елементарною ланкою мається на увазі частина об'єкта, системи управління і т.д., яка реалізує елементарну функцію. Структурна схема для розробки, яка включена у WordPress виглядає наступним чином:

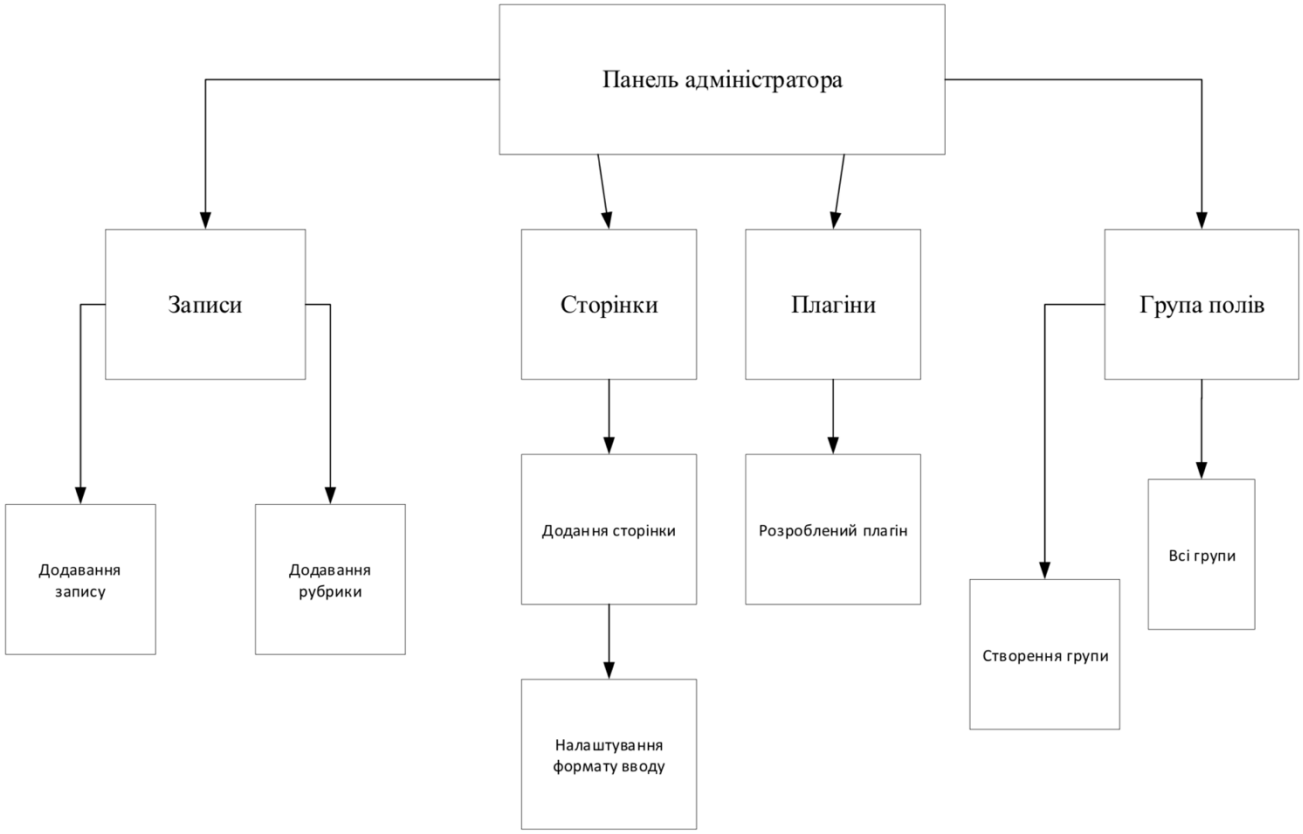


Рисунок 4.1 – Структурна схема плагіну

Плагіна, який розробляється в магістерській дисертації повинен бути частиною системи керування вмістом WordPress. Структурна схема починається з демонстрації структури панелі адміністратора. Тут є основні розділи, які відносяться до плагіну: записи, сторінки, плагіни та группа полів.

Перш за все треба звернути увагу на те, що спочатку адміністратора переходить до розділу «Плагіни» та знаходить створене в диссертації розширення, після чого його треба активувати. Активація та деактивація плагіна WordPress здійснюється з використанням хуків. Вони забезпечують виконання PHP дій, коли плагін активується або деактивується.

При активації як правило потрібно запустити процедуру встановлення плагіну, щоб, наприклад:

- додати опції (налаштування) за замовчуванням;
- встановити таблиці бази даних;
- додати правила ЧПУ;
- створити директорії;
- при деактивації зазвичай очищають тимчасові дані:
- кеш в тимчасових опціях;
- файли у тимчасових каталогах.

Крім активації і деактивації, важливим моментом є видалення плагіна. Підключитися до цього процесу можна через функцію register_uninstall_hook().

Якщо плагін записує будь-які дані куди завгодно: в опції, базу даних, файли і т.д., то при повному видаленні плагіна, логічно видалити і всі дані які відносяться тільки до нього, щоб вони не завантажували сайт в цілому.

Процес видалення плагіна запускається, коли користувач відключив плагін, а потім натиснув на посилання «Видалити» в панелі адміністратора на сторінці плагінів.

При деактивації повинні видалятися (очищатися) тільки тимчасові дані, а всі дані тільки при видаленні плагіна. Крім того, коли плагін видаляється, хук деактивації вже спрацював. Тобто активний плагін завжди спочатку деактивується і тільки потім видаляється – неможливо видалити плагін, якщо він активний.

Перед видаленням плагіна (посилання «Видалити» на сторінці плагінів), WordPress виконує деякі дії, завдяки яким можна видалити всі сліди перебування плагіна на сайті. Переваги використання такого підходу в тому, що не потрібно піклуватися про

попередження користувачів про видалення слідів плагіна, перед видаленням самого плагіна. Це свого роду API видалення плагінів – WordPress робить частину роботи за користувача.

Після активації плагіна користувач переходить відразу в розділ «Группа полів», а далі «Створення группи». Саме тут йому для початку потрібно створити групу полів. Йдеться про довільні поля, в які в подальшому буде записуватися контент, а потім з їх допомогою можна буд редагувати і виводити группу полів в потрібному форматі. Все це реалізовано за допомогою призначеного для користувача інтерфейсу.

Щоб плагін працював коректно його в будь-якому випадку доведеться налаштувати. Створення групи полів – це перший крок, після нього потрібно перейти до другого і вибрати цільову сторінку. Адміністратор переходить в розділ «Сторінки», далі «Додавання сторінки» та знаходить блок «Налаштування формату виводу».

Можна не тільки створювати нову сторінку, а ще і вивести контент на вже існуючу в WordPress. Як тільки обидва налаштування будуть зроблені можна переходити в розділ «Записи». Спочатку створюємо рубрику у розділі «Додавання рубрики», потім створюємо записи у розділі «Додавання запису». Важливо щоб при створенні запису адміністратор обрав рубрику для нього, до якої відноситься группа полів, що створили раніше.

Тепер користувачеві потрібно додати необхідний контент в групу довільних полів. Це може бути текст, список або картинка. В процесі розробник плагін буде розширюватися і приймати інші типи контенту. До кроку додавання контенту користувач зможе повернутися в будь-який час, тому що тут же відбувається і редагування. Це не статичні довільні поля, які не можна змінити. Все що з'являється на сайті можна легко міняти через панель адміністратора WordPress, що спрощує взаємодію власника і веб-ресурсу.

Далі необхідно перейти на цільову сторінку, яка була раніше обрана в налаштуваннях. Плагін працює таким чином, що людина спочатку створює групу полів, вибирає цільову сторінку і потім вони вже з'являються на ній в панелі адміністратора.

Коли користувач знаходить потрібну сторінку в адміністративній панелі, він може задати формат виводу. Це ще одна особливість плагіна, яка дозволяє вибрати вид відображення контенту на сайті. Під форматом мається на увазі вибір конкректного шаблону. Наприклад, якщо мова йде про текст з кількох абзаців, то їх можна розмістити різними способами. Це можуть бути як повні абзаци один за одним, так і можливість розмістити, скажімо, 2 абзаци поруч в 2 колонки. Така функція плагіна допомагає змінювати вид верстки вручну через панель адміністратора і позбутися від послуг верстальника. Згодом форматів виведення контенту буде більше, що допоможе вирішувати будь-які завдання власнику сайту.

Коли на цільовій сторінці в WordPress користувач визначився з форматом виведення, він може ще раз перевірити контент. Швидше за все буде також реалізована можливість редагування даних безпосередньо на цільовій сторінці. Користувачеві після всіх цих маніпуляцій залишається тільки натиснути кнопку «Опублікувати». Після цього WordPress запам'ятає зміни і виведе контент в потрібному місці і в обраному форматі.

На даний момент структурна схема демонструє як виглядає панель адміністратора WordPress разом з розробленим плагіном після його активації.

5 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

WordPress – система керування вмістом сайту з відкритим вихідним кодом. CMS написана на мові програмування php з використанням баз даних MySQL. Сьогодні використовується для створення сайтів різної складності, від простих візиток до інтернет-магазинів.

У WordPress вбудована система тем і плагінів, які ідеально увійшли в архітектуру CMS. Завдяки таким доповненням можна розширювати функціонал сайтів так, як захоче розробник. Використання WordPress спрощує роботу з готовим сайтом і допомагає прискорити процес розробки.

Оновлений рейтинг популярності CMS в світі і сегментах Європа, Україна, Росія. Рейтинг складений на основі зведених даних декількох незалежних аналітичних ресурсів – iTrax, Builtwith, RatingRuneta, CMS Magazine і IT-rating станом на березень 2018.

%	Ukraine	Russian	Word
WordPress	41,45	36,59	53
Joomla	13,58	12,19	5
Drupal	4,23	4,42	1
DLE	1,06	1,44	1
Bitrix	5,53	20,81	0,5
Adobe Muse	0,50	1,02	0,5
Wix	0,50	1,80	0,5
OpenCart	7,63	2,67	1
MODX	0,50	4,81	0,5
uCoz	0,50	1,30	0,5
UMI	0,82	1,45	0
NetCat	0,06	1,11	0
CS-Cart	0,12	0,67	0
Other	33,17	21,74	38,50

Рисунок 5.1 – Статистика використання CMS за 2018 рік

В цілому показники поширення CMS сформувалися ще кілька років тому і до сих пір дані особливо не змінюються. Серед безкоштовних CMS великим впливом користується WordPress. Решта системи втрачають частку, яка перетікає від одних

хмарних платформ до інших.

У сфері масового сегмента комерційних CMS теж немає помітних коливань (вони і менш вірогідні, ніж в безкоштовних).

Чому ж люди вважають за краще використовувати WordPress?

- універсальність. Ця CMS, допомагає реалізувати величезну кількість різноманітних проектів, з урахуванням установки плагінів. При цьому, у безкоштовної системи управління вмістом можуть виникнути проблеми зі складними сайтами, але якщо постаратися, від них можна позбутися і все якісно оптимізувати.
- модулі. У репозиторії плагінів для WordPress тисячі варіантів, що допомагають налаштовувати SEO, міняти візуальний редактор, контролювати статистику відвідувань, створювати кастомні типи записів і багато іншого. Щоб встановити плагін досить одного кліка і 5 хвилин для налаштування, а деякі навіть не треба налаштовувати.
- простота. WordPress відрізняється простим інтерфейсом. З цієї причини працювати з платформою легко і приємно, потрібні функції можна знайти на чисто інтуїтивному рівні. Крім того, завдяки простоті налаштувань можна швидко адаптувати зовнішній вигляд веб-ресурсу і його функціонал під свої потреби.
- функціонал. Код WordPress у відкритому доступі, тому CMS можна легко розширювати на власний россуд. Програміст може легко додати якісь можливості або навпаки прибрати зайве, щоб знизити споживання ресурсів системи і прискорити сайт.
- оптимізація. Цей пункт важливий для тих користувачів, які прагнуть просувати свій ресурс в найвідоміших пошукових системах. Наприклад, Google добре реагує на сайти на основі цієї CMS, значно краще, ніж на інші.
- популярність. Вордпресс дуже простий у використанні і тому легко знайти навчальні уроки навіть в YouTube. Крім цього випущено ще й безліч літератури

по темі розробки сайтів та управлінню преоктом на правах адміністратора. Перш за все це корисно власникам сайту на WordPress, які замовили розробку і після цього через панель адміністратора можуть вирішувати будь-які завдання самостійно не залучаючи знову програмістів.

Доступність. Ця CMS доступна, за неї не потрібно платити гроші, до того ж, її інтерфейс переведений на багато мов, в тому числі і на українську.

У WordPress є і кілька недоліків:

- вимогливість. Система управління вмістом споживає дуже багато ресурсів хостингу, щоб справно працювати. Відповідно, складні проекти, де потрібна велике споживання пам'яті можуть працювати повільно.
- захист. Оскільки WordPress безкоштовна система, то і над захистом від злому розробники працюють слабо. Таким чином, це досить вразлива платформа, яка не підійде для серйозних проектів на кшталт онлайн обмінників валюти.
- індексація. На сьогоднішній день WordPress відмінно індексується у всіх відомих пошукових системах, але в тих що менш популярні сайти на CMS часто не приймаються. Якщо ваш сайт розрахований на західний ринок, де величезна кількість різноманітних пошукових систем, то скоріше за все просунути його для іноземців буде дуже складно.

Які переваги в розробці на WordPress?

- глобальні змінні. Так, є певні складнощі з пошуком помилок в різних частинах проекту. Однак що може бути простіше, ніж використання глобальних змінних? Необхідність проходу змінної через різні функції може ускладнити розуміння новачка про те, що відбувається всередині коду, точно, як і нерозуміння, звідки беруться ці змінні. У нашому ж випадку вони загальновідомі, так як описані в заздалегідь відведеному місці і представлені як належне.
- простір роботи з базою даних. У WordPress вся концепція MVC завуальована за шаблонними запитами. З одного боку – це може здатися незрозумілим тим хто

звик працювати з фреймворками, але на ділі виявляється дуже зручно в процесі розробки. Наприклад, робити запит до БД через post_type якщо потрібно дістати з неї якусь інформацію. Комусь може здатися незвичним, що роблячи запит він може вплинути на подальше виведення контекстної інформації, але якщо зрозуміти, як цього уникнути, то проблема відразу зникає.

- файлова архітектура. Структура файлів ядра WordPress і кожної окремої теми зроблена максимально зручно. Інтуїтивно зрозуміле іменування папок і файлів швидко стане зрозумілим і дасть можливість розширювати систему керування вмістом. При бажанні навіть можна самостійно рознести все по класах і зробити подобу архітектури MVC і все буде працювати так, як того захоче розробник. Також є можливість використовувати less і sass.
- робота з зображеннями. Не менш важливий плюс, ніж всі перераховані вище. Система автоматично обжимає зображення, щоб не навантажувати сайт і створює кілька варіантів з різними розмірами. Таким чином, одну картинку можна використовувати в різних місцях, наприклад, як основне зображення всередині одного запису з конкретним розширенням і одночасно в якості мініатюри на сторінці всіх записів уже в іншому розширенні.

5.1 Теми в WordPress

Тема для WordPress – це оболонка з готовим оформленням для вашого сайту. На ділі це все-таки куди більше, ніж просто зовнішній вигляд для проекту. Тема може забезпечувати повний контроль над контентом на сайті з можливістю його додавання, видалення і редагування прямо з панелі адміністратора.

Тема являє собою набір файлів, які разом дозволяють створити оформлення графічного інтерфейсу і вмісту сайту. Представлені всередині теми файли називають шаблонами. Готові, безкоштовні теми зі сховища WordPress дозволяють змінити зовнішній вигляд сайту без втручання в код. Якщо розглянути ієрархію файлів всередині папки з темою, то ви побачите зображення (.jpg, .gif), каскадні таблиці стилів (.css), що

налаштовують сторінки, а так само різноманітні скрипти (.php, .js). Основну інформацію про шаблони теми можна дізнатися з панелі адміністратора або прочитати у файлі index.php.

Наприклад, у вас є сайт, де ви пишете про техніку і там може бути розділ зі смартфонами і комп'ютерами. На ділі це різні теми і вони потребують якесь оригінальне оформлення. За допомогою однієї теми можна створити 2 різні рубрики: «смартфони» і «комп'ютери». Крім того, що наявність рубрик допоможе розділити інформацію за двома розділами у вас також з'явиться можливість по-різному їх оформити. Найголовніше, що все це робиться з панелі адміністратора.

Через панель адміністратора є можливість встановлювати готові теми зі сховища WordPress. Всі теми зберігаються в розділі «Консоль» – «Зовнішній вигляд» – «Теми». Для того, щоб сайт перевтілився, використовуючи оформлення вашої теми досить натиснути біля неї на кнопку «Активувати». Як правило, ніяких додаткових налаштувань не потрібно, вам досить просто перейти на сайт і побачити зміни.

Оскільки в магістерській дисертації створюється плагін для динамічного виведення контенту в темах, які робляться з нуля, то варто розглянути «анатомію» теми.

Теми WordPress знаходяться в піддиректорії wp-content/themes/. Директорія теми містить таблиці каскадних стилів, файли шаблонів, файл додаткового функціоналу (functions.php) і картинки. Наприклад, якщо створити нову тему під назвою «theme1», то розмістити її треба буде за адресою wp-content/themes/theme1/. Щоб краще зрозуміти ієрархію файлів теми варто подивитися одну зі стандартних тем, які є в WordPress за замовчуванням після установки: «Twenty Fifteen», «Twenty Sixteen» і «Twenty Seventeen».

Усередині кожної теми є 3 типи файлів. Перший – таблиця стилів під назвою style.css, яка відповідає за зовнішній вигляд сайту. Другий є файлом, який поєднує весь функціонал – functions.php. Третій – це файли з розширенням .html, які є шаблонами і визначають в якому вигляді буде виводитися інформація на сайті.

Якщо говорити про файли стилів, то їх може бути кілька. Хорошим тоном розробки вважається, як мінімум, поділ основних стилів для стандартних моніторів і медіазапитів, які допомагають адаптувати сайт під мобільні пристрої. При бажанні можна створювати скільки завгодно файлів стилів з поділом коду під конкретні ділянки сайту, а потім зберігати їх в папці «css».

В обов'язковому порядку всередині теми повинен бути файл function.php. Усередині нього містяться необхідні функції для роботи теми і підключення ключових бібліотек. Файл працює подібно плагіну, і якщо він присутній в каталозі теми, яку ви використовуєте, то він автоматично завантажується під час ініціалізації WordPress. Це працює, як для сторінок панелі адміністратора, так і для інших, зовнішніх сторінок.

Як правило, всередині function.php розміщують підключення до файлів стилів і скриптів на JavaScript. У підсумку виходить, що в шаблонах код з підключенням вже не потрібен і разом з цим всі необхідні файли будуть довантажувати автоматично коли користувач зайде на сайт або власник ресурсу зайде в панель адміністратора.

Шаблони – це файли з розширенням .php, які генерують сторінки сайту або їх частини за запитом користувача. При створенні своєї теми можна придумати різноманітні варіанти шаблонів і скомбінувати все це в готовий ресурс.

CMS WordPress дозволяє використовувати різні файли шаблонів для генерації кількох видів сторінок. Це ефективно якщо у вас різні розділи і кожен з них ви хочете відображати в своєму унікальному стилі. Ніхто не забороняє створювати і один шаблон для всього сайту, наприклад, в index.php. В такому випадку на різних сторінках контент відрізнятиметься, але зовнішній вигляд в цілому залишиться однаковим.

Серед основних файлів, які формують сайт використовуються такі шаблони:

- header.php – шапка сайту, де крім меню і подібних елементів в самій розмітці міститься підключення посилань або скриптів при необхідності;
- footer.php – так званий підвал сайту, де знаходиться його логічне завершення і можливо підключається скрипт для скидання статистики по відвідинам;

- sidebar.php – якщо сайт зроблений з додатковою областю праворуч або ліворуч, то вона називається sidebar і вся її розмітка розміщується в цьому файлі і потім легко виводиться на потрібних сторінках;
- comments.php – шаблон для реалізації зовнішнього вигляду коментарів, він повинен бути завжди, адже спочатку WordPress був системою тільки для блогів, де коментарі одна з найважливіших частин, при необхідності їх можна прибрати;
- single.php – шаблон для опису розмітки запису;
- page.php – шаблон для опису розмітки однієї сторінки.

Наприклад, якщо потрібно в одному шаблоні з'єднати всі складові сайту, то на самому початку потрібно викликати функцію <? php get_header (); ?>, а в самому кінці <? php get_footer (); ?>. Якщо у вас є додаткова область з контентом, то просто викличте функцію <? php get_sidebar (); ?>. Саме за рахунок поділу сторінок сайту на окремі модулі і комбінуванні їх в потрібному порядку на конкретних шаблонах, WordPress і є зручною системою управління вмістом.

5.2 Плагіни

Плагіни для WordPress – це доповнення, написані розробниками для того, щоб розширити функціонал CMS. З технічного боку плагін, як і тема для WordPress, складається з набору файлів .php, всередині яких міститься програмний код, який додає нові можливості.

На даний момент WordPress став дуже потужною платформою для реалізації сайтів різної складності і багато в чому за рахунок плагінів. Одного такого доповнення може бути досить, щоб, наприклад, реалізувати тестування користувачів на сайті і тим самим перетворити блог в ресурс по вивченню мов або якихось наук. Знайти в офіційному репозиторії можна тисячі плагінів і кожен з них додає якісь нові функції.

У стандартній версії WordPress плагіни не встановлені, незважаючи на те, що вони дуже корисні. Справа в тому, що такі розширення займають чимало місця і при цьому споживають ресурси хостингу. Відповідно, встановлювати плагіни потрібно з розумом,

інакше вони перенавантажуватимуть сайт і працювати з ним буде неможливо. В основному, творці таких доповнень для системи керування вмістом намагаються оптимізувати свій продукт, але зробити це правильно виходить у одиниць.

Оскільки для встановлення плагіна досить перейти в однойменний розділ в панелі адміністратора і натиснути кнопку «Активувати» під потрібним розширенням, ви можете в будь-який момент поставити те що вам потрібно. Пошук плагінів з офіційного репозиторія здійснюється безпосередньо з панелі адміністратора або на офіційному сайті WordPress. Також ви можете скачати плагін у вигляді ієрархії файлів і потім за допомогою FTP-доступу залити їх на хостинг і вже потім активувати через панель адміністратора.

Будь-який плагін створюється на мові програмування PHP, з використанням API WordPress. Завданням будь-якого плагіна має бути створення нового функціоналу з мінімальним можливим навантаженням, сумісність з ядром WordPress і грамотна робота коду в умовах постійного оновлювання системи.

Першим кроком в розробці плагіна є створення окремого каталогу (папки) для вмісту плагіна, наприклад: test-plugin. У цій папці будуть всі файли плагіна. Серед них особливе місце займає головний файл плагіна, який бажано, повинен збігатися з назвою самої папки плагіна, наприклад test-plugin.php. У результаті повинна вийти така структура: wp-content/plugins/test-plugin/test-plugin.php.

Після створення файлу плагіна, в нього потрібно помістити заголовки, щоб WordPress розпізнав плагін як плагін. Плагін може бути особистий (створюється тільки для одного сайту), а може бути публічний (для загального користування, викладається в репозиторій плагінів WordPress). На даний момент функціонал розроблений в магістерській дисертації є особистим і використовується виключно для реалізації проектів автора. Надалі планується розширення плагіна і додавання його в офіційний репозиторій.

Вимоги до особистого плагіну, зазвичай мінімальні, а ось з публічним все складніше, потрібна ліцензія, підтримка, сумісність, локалізація та інше. Тому публічні плагіни створювати в рази складніше.

Ліцензія повідомляє користувачів, як вони можуть використовувати код плагіна в своїх цілях. Для підтримки сумісності з ядром WordPress рекомендується вибрати ліцензію, що працює з GNU General Public License (GPLv2 +).

Щоб не створювати файли і структуру з нуля, можна використовувати шаблон для створення плагіна – WordPress Plugin Boilerplate[4]. Але використовувати потрібно генератор цього шаблону, де вказується назва плагіна, яка буде використана в назвах папок, класів і функцій – WordPress Plugin Boilerplate Generator. Саме такий підхід і використовувався в магістерській дисертації. Суть в тому, щоб згенерувати правильну структуру і потім вже працювати з нею. У даній ситуації структура адаптувалася під патерн MVC для підвищення зручності та ефективності розробки.

Стандартний шаблон[5] являє собою організовану об'єктно-орієнтовану основу, з можливістю розширення. Також шаблон дотримується стандартів PHP коду для WordPress. Використовуючи такий підхід, можна бути впевненим в більш чіткій і зрозумілій структурі нового плагіна.

По всьому ядру WordPress розташовано безліч хуків. Хуки[6] дозволяють підключитися в певних місцях до коду ядра WordPress, щоб змінити його поведінку, при цьому не зачіпаючи файли ядра. Це такі собі функції виклику конкретної події в потрібному місці.

Існує два типи хуків в WordPress:

- події (actions);
- фільтри (filters).

Події дозволяють додавати або змінювати функціонал WordPress, в той час як фільтри дозволяють змінювати дані або рядки. Хуки потрібні не тільки для розробників плагінів – вони використовуються всюди: в самому ядрі WordPress, в плагінах і темах. Саме хуки роблять WordPress таким гнучким.

Безпосередньо до плагінів відносяться три функції відповідаючі за хуки плагіна:

- register_activation_hook() – реєструє функцію, яка буде спрацьовувати під час активації плагіна. Використовується для додавання налаштувань плагіна і т.п;
- register_deactivation_hook() – реєструє функцію, яка повинна запускатися після деактивації плагіна. Використовується для видалення тимчасових даних плагіна;
- register_uninstall_hook() – реєструє функцію, яка викликається при видаленні плагіна. Використовується при видаленні плагіна для видалення всіх даних плагіна: в налаштуваннях, в файлах, в базі даних і т.д.

Всі ці хуки використовувалися в магістерській дисертації для коректної роботи плагіна всередині екосистеми WordPress.

5.3 Довільні поля

У WordPress існують такі інструменти для роботи з плагінами і темами, як довільні поля[7]. Вони зустрічаються дуже часто і дозволяють додавати, видаляти і редагувати певний контент на сайті. Поля є частиною ядра WordPress і можуть бути як видимими, так і невидимими. Іноді вони використовуються не явно, так що ви можете навіть не підозрювати, що вони є на сайті.

За замовчуванням довільні поля не видно, але кожен користувач системи управління вмістом може легко налаштувати їх відображення. Досить викликати в панелі адміністратора допоміжне меню «Налаштування екрану», яке розташовується в самому верху. Далі в розділі «На екрані відображаються» відзначте галочкою «Довільні поля» і їх можна буде побачити в записах і сторінках.

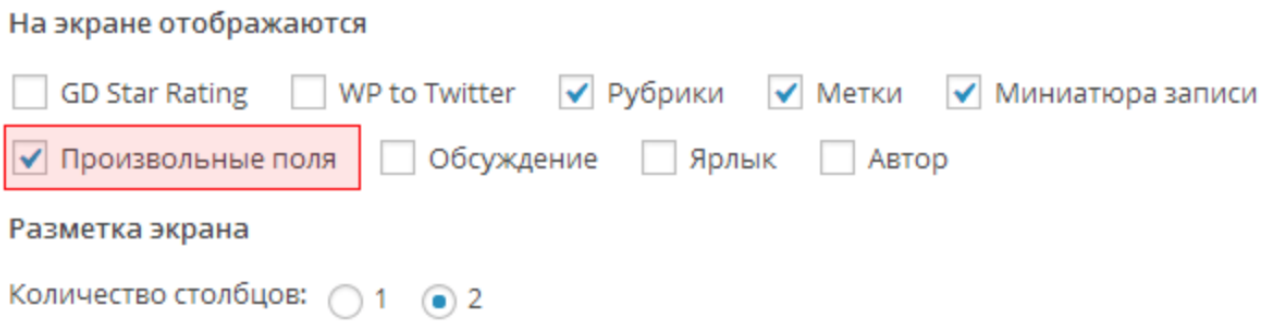


Рисунок 5.2 – Меню «Налаштування екрану»

В записах та на сторінках довільні поля мають наступний вигляд:

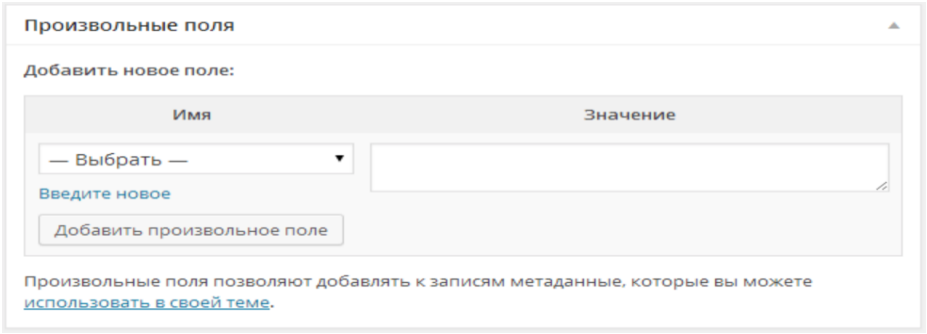


Рисунок 5.3 – Відображення довільних полів

Залежно від плагінів і тем, які встановлені на вашому сайті, меню, що випадає в колонці «ім'я» вже може містити якісь довільні поля. Але якщо сайт новий, то і полів може не бути.

Перед створенням нового довільного поля давайте подивимося на те, що собою представляють ці поля і як можна їх використовувати. У кожного поля є 2 частини: ім'я поля і його значення. У значенні написаний контент. Також там можна вказати і веб-адреси поряд зі звичайним текстом.

Використання довільних полів можна розлянути на прикладі сайту, присвяченому книгам. Частина інформації, яку можна додати до основного посту, можна доповнити авторською інформацією.

Для сайту про книги можна додати довільне поле для відображення даних за найсвіжішими виданнями під назвою «останні книги». Для створення нового поля просто клікаємо на кнопку «Додати» і заповнюємо назву поля, а далі натискаємо на кнопку додавання нового поля «Додати довільне поле».

Наприклад, можна також включити такі довільні поля:

– для блогу про їжу – «нові рецепти»;
– для сайту про музику – «популярні треки»;
– для блогу про покупках – «останні придбання»;
– для сайтів з оглядами – «рейтинг постів».

Будь-яку інформацію, що цікавить читачів додатково можна включити в довільне поле.

Після створення довільного поля можна дати можливість кожному вашому користувачу, який пише пости в блозі, додати нову інформацію, доступну з меню, що випадає. Така інформація буде доступна для підстановки додаткових даних у всіх нових постах.

Довільні поля відображаються на екранах в розділі «Додати новий запис» або «Редагувати запис». Безпідставно до поля можна привязати різні значення в залежності від того, яку функцію виконує поле. Також можна зробити їх обов'язковими полями для публікації поста (заповнювати, перш ніж опублікувати). Це спрощує роботу з сайтом з боку модератора, особливо якщо їх на одному ресурсі кілька.

У WordPress легко створювати довільні поля через панель адміністратора і заповнювати їх, але що робити далі? Після того, як вони були створені, необхідно вивести дані. Робиться це безпосередньо в шаблоні за допомогою додавання відповідного коду. WordPress може виводити вміст кожного довільного поля за допомогою функції по типу <?php the_test (); ?>, де test – це назва конкретного довільного поля.

Звідси можна зробити висновок, що кожне довільне поле можна виводити викликаючи функцію з його назвою. Найголовніше, що розташувати цей код можна в

будь-якому місці шаблону і туди буде виведений контент поля. Крім тексту, за замовчуванням, інших варіантів не має.

На даний момент в WordPress довільні поля в своєму звичайному вигляді реалізовані дуже незручно. Саме тому розробники завжди користуються плагіном Advanced Custom Fields. За останні 5 років, він став мало не стандартним доповненням до системи керування вмістом. Перевага в тому, що він дозволяє зручніше створювати довільні поля через панель адміністоратора, додавати їх до конкретних записів або сторінок, а також має готову базу, відразу з великою кількістю різновидів полів. Все це є вже в безкоштовній версії, але при необхідності можна придбати і платну, де можливості ще більше розширені в сторону довільних полів для вирішення дуже специфічних завдань.

Працюючи з плагіном ACF і прийшла ідея створити своє рішення для магістерської дисертації, яке буде зручніше, просте в використанні і матиме новий функціонал. На жаль, в ACF немає можливості динамічно міняти відображення контенту, а це часто дуже важлива річ, особливо для власника сайту, який намагається редагувати його власними силами. Саме тому за основу розробленого в магістерській дисертації плагіна були взяті довільні поля і оброблені таким чином, щоб вийшло динамічно виводити контент з панелі адміністратора.

6 ОПИС СХЕМИ БАЗИ ДАНИХ

Оскільки бази даних[8] для сайтів використовуються з метою зберігання різної інформації і, спрощено, представляють собою деякий набір взаємозв'язаних таблиць, необхідно вибирати інструменти для їх проектування, які допоможуть сайту працювати максимально швидко. Розміри таблиць в базі даних різні, їх кількість довільна. Саме в базі даних зберігається на сервері необхідна інформація для роботи плагіну, наприклад, назви довільних полів, список шаблонів, типи полів і т.д.

Для розробки плагіну, який динамічно виводить контент використовувався інструмент міграцій, який значно спрощує взаємодію проекту з базою даних і прискорює побудову таблиць. Таким чином, запит йде безпосередньо до всієї бази і постійно змінюється при появі в середині плагіну нової моделі або контролера. Завдяки такому підходу вдалося прискорити працездатність сайту у звязці з плагіном, що в даній ситуації є дуже важливою складовою, адже сайт может містити багато інформації і у WordPress можуть з’явитися проблеми зі швидкістью якщо до нього підключено багато не оптимізованих плагінів.

Після додавання контексту даних у вигляді сервісу можна надалі отримувати його в конструкторі контролера через механізм впровадження залежностей. Визначивши всі налаштування можна використовувати міграцію для створення бази даних.

Міграція баз даних використовується в роботі плагіну, коли може виникнути ситуація, де модель змінюється. Протягом розробки проекту кілька разів доводилося додавати нові властивості. При цьому вже є база даних, де зберігаються певні дані. Щоб оновити їх без видалення і створення нових таблиць використовується механізм MVC міграцій.

Міграції дають можливість просто додавати, по суті нові властивості при необхідності. Оскільки робота з базою даних йде через контекст даних. Це значно спрощує процесс розробки, навідміну від інших CMS, де необхідно постійно оновлювати вміст бази данних за допомогою різних розширень. Також використання міграцій у проекті допомогло позбудится кешування сторінки, яке навіть після

оновлення не показує зміни на сайті або робить певний функціонал не працюючим для конкретного користувача в певний момент використання системи. Це було необхідною мірою оскільки плагін виводить контент динамічно, а отже повинен працювати без кешування.

В контексті необхідно додавати нові властивості в ситуації, коли в проекті з'являються нові моделі. Коли змінюється контекст даних необхідно привести міграцію від старої схеми бази даних до нової. Для цього достатньо створити нову міграцію. За допомогою конфігурацій можна створювати нові таблиці в базі даних і поля для них або вже існуючих таблиць.

WordPress використовує кілька взаємопов'язаних таблиць. Між ними встановлені зв'язки один до багатьох. Наприклад, до однієї сторінки може бути багато коментарів. Наведена нижче діаграма взята з кодексу WordPress. На ній показані таблиці і зв'язку між ними:

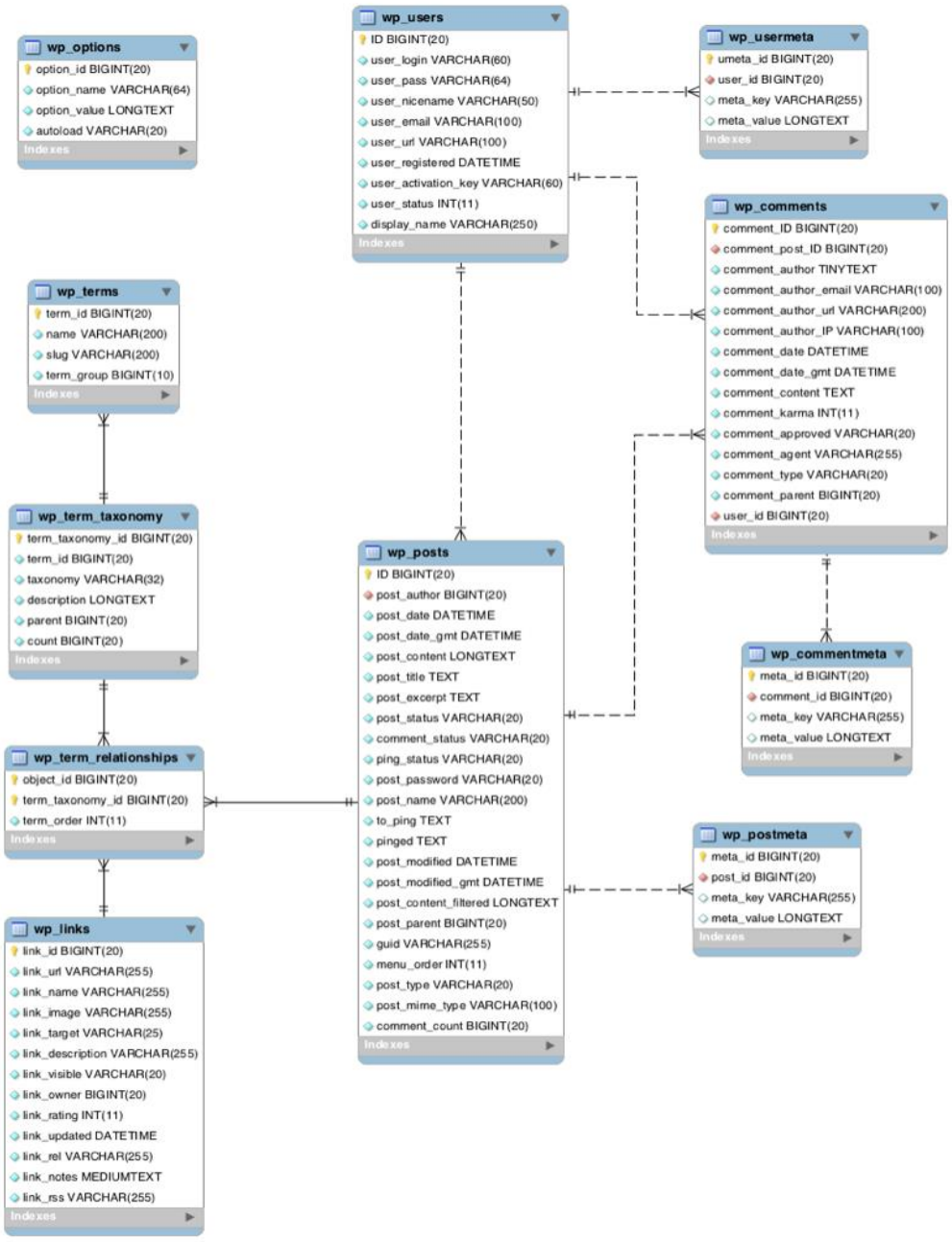


Рисунок 6.1 – Схема бази даних для блогу

Більшість таблиць пов'язані з однією або декількома іншими з допомогою одного поля. Це поле буде унікальним ідентифікатором для кожного запису (приклад post_id). Більш докладно для кожної таблиці:

Таблиця 6.1 – Структура бази данних для блогу

Назва таблиці	Данні	Зв'язки з іншими таблицями
wp_posts	Записи, сторінки, вкладення, редакції, призначені для користувача записи	wp_postmeta через post_id wp_term_relationships через post_id
wp_postmeta	Метадані записів, сторінок і т.д.	wp_posts через post_id
wp_comments	Коментарі	wp_posts через post_id
wp_commentmeta	Метадані коментарів	wp_comments через comment_id
wp_term_relationships	Зв'язки між таксономією і записами, сторінками і т.д.	wp_posts через post_id wp_term_taxonomy через term_taxonomy_id
wp_term_taxonomy	Таксономія (включаючи категорії і мітки)	wp_term_relationships через term_taxonomy_id

wp_terms	Ваші категорії, мітки і терміни призначені для користувача таксономій	wp_term_taxonomy через term_id
wp_links	Посилання в вашому блоці (як правило, зараз не використовується)	wp_term_relationships через link_id
wp_users	Користувачі	wp_posts через post_author
wp_user_meta	Метадані для кожного користувача	wp_users через user_id
wp_options	Опції і налаштування сайту (встановлюються в панелі адміністратора на сторінці налаштувань і в темах/плагінах)	Відсутні

Варто відзначити кілька речей:

- таблиці бази даних за замовчуванням мають префікс wp_. Ви можете його змінити (наприклад, при встановленні);
- таблиця wp_posts є найбільш важливо. Саме в ній зберігатися більшість даних;
- тільки одна таблиця не пов'язана з іншими – таблиця wp_options. У ній зберігаються дані про сайт і налаштування WordPress, які не мають відношення до записів або користувачам;
- у таблицях wp_users і wp_comments дані не пов'язані. В налаштуваннях WordPress можна вказати, що тільки зареєстровані користувачі можуть

залишити коментар. Не дивлячись на це, WordPress не зберігає зв'язку про коментарі і користувача, який їх відправив;

– більшість таблиць пов'язані з однією або декількома іншими за допомогою одного поля. Це поле буде унікальним ідентифікатором для кожного запису (приклад post_id).

В плагіні робота проходить зі стандартною базою данних WordPress. Деякі таблиці зв’язані з роботою плагіну. Наприклад, таблиця wp_posts виглядає так:

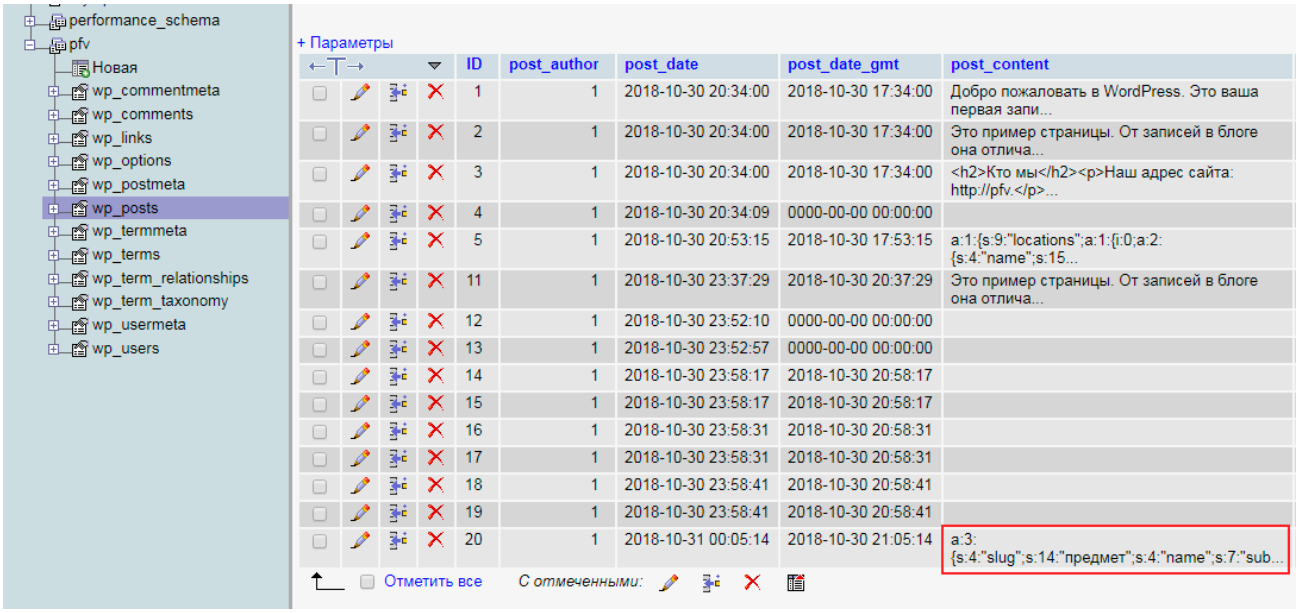
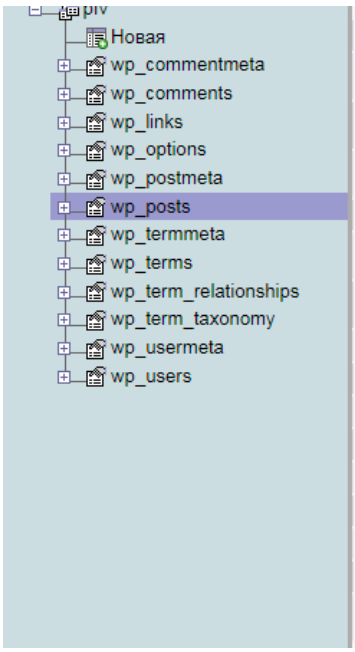


Рисунок 6.2 – Таблиця wp_posts



post_title	post_excerpt	post_status	comment_status	ping_status	post_password
Привет, мир!		publish	open	open	
Пример страницы		publish	closed	open	
Политика конфиденциальности		draft	closed	open	
Черновик		auto-draft	open	open	
Преподаватели		publish	closed	closed	
Пример страницы		inherit	closed	closed	
Черновик		auto-draft	open	open	
Черновик		auto-draft	open	open	
Препод 1		publish	open	open	
Препод 1		inherit	closed	closed	
Препод 2		publish	open	open	
Препод 2		inherit	closed	closed	
Препод 3		publish	open	open	
Препод 3		inherit	closed	closed	
pfv_field_rc0n9l		publish	closed	closed	

Рисунок 6.3 – Таблиця wp_posts

post_name	to_ping	pinged	post_modified	post_modified_gmt	post_content_filtered
%d0%bf%d1%80%d0%b8%d0%b2%d0%b5%d1%82-%d0%bc%d0%b8%...			2018-10-30 20:34:00	2018-10-30 17:34:00	
sample-page			2018-10-30 23:59:53	2018-10-30 20:59:53	
privacy-policy			2018-10-30 20:34:00	2018-10-30 17:34:00	
			2018-10-30 20:34:09	0000-00-00 00:00:00	
%d0%bf%d1%80%d0%b5%d0%bf%d0%be%d0%b4%d0%b0%d0%b2%d...			2018-10-31 02:25:46	2018-10-30 23:25:46	
2-revision-v1			2018-10-30 23:37:29	2018-10-30 20:37:29	
			2018-10-30 23:52:10	0000-00-00 00:00:00	
			2018-10-30 23:52:57	0000-00-00 00:00:00	
%d0%bf%d1%80%d0%b5%d0%bf%d0%be%d0%b4-1			2018-10-30 23:58:57	2018-10-30 20:58:57	
14-revision-v1			2018-10-30 23:58:17	2018-10-30 20:58:17	
%d0%bf%d1%80%d0%b5%d0%bf%d0%be%d0%b4-2			2018-10-30 23:59:06	2018-10-30 20:59:06	
16-revision-v1			2018-10-30 23:58:31	2018-10-30 20:58:31	
%d0%bf%d1%80%d0%b5%d0%bf%d0%be%d0%b4-3			2018-10-30 23:59:16	2018-10-30 20:59:16	
18-revision-v1			2018-10-30 23:58:41	2018-10-30 20:58:41	
pfv_field_rc0n9l			2018-10-31 00:05:14	2018-10-30 21:05:14	

Рисунок 6.4 – Таблиця wp_posts

post_parent	guid	menu_order	post_type	post_mime_type	comment_count
0	http://pfv/?p=1	0	post		1
0	http://pfv/?page_id=2	0	page		0
0	http://pfv/?page_id=3	0	page		0
0	http://pfv/?p=4	0	post		0
0	http://pfv/?post_type=pfv_field_groups&p=5	0	pfv_field_groups		0
2	http://pfv/2018/10/30/2-revision-v1/	0	revision		0
0	http://pfv/?p=12	0	post		0
0	http://pfv/?p=13	0	post		0
0	http://pfv/?p=14	0	post		0
14	http://pfv/2018/10/30/14-revision-v1/	0	revision		0
0	http://pfv/?p=16	0	post		0
16	http://pfv/2018/10/30/16-revision-v1/	0	revision		0
0	http://pfv/?p=18	0	post		0
18	http://pfv/2018/10/30/18-revision-v1/	0	revision		0
5	http://pfv/pfv_field/pfv_field_rc0n9l/	0	pfv_field		0

Рисунок 6.5 – Таблиця wp_posts

Опис ключових полів:

– post_content – якщо дане поле відноситься до post_type = «pfv_field_groups», то в нього зберігаються дані місця розташування групи полів у вигляді серіалізованого масиву, якщо дане поле відноситься до post_type = «pfv_field», то в нього зберігаються дані поля (ярлик, машинне ім'я, тип поля);

– post_name – якщо дане поле відноситься до post_type = «pfv_field», то в нього зберігається: pfv_field (унікальний ключ);

– post_parent – якщо дане поле відноситься до post_type = «pfv_field», то в нього зберігається id групи полів до якого відноситься дане поле;

– post_type – типи записів.

В плагіні таблиця wp_postmeta виглядає так:

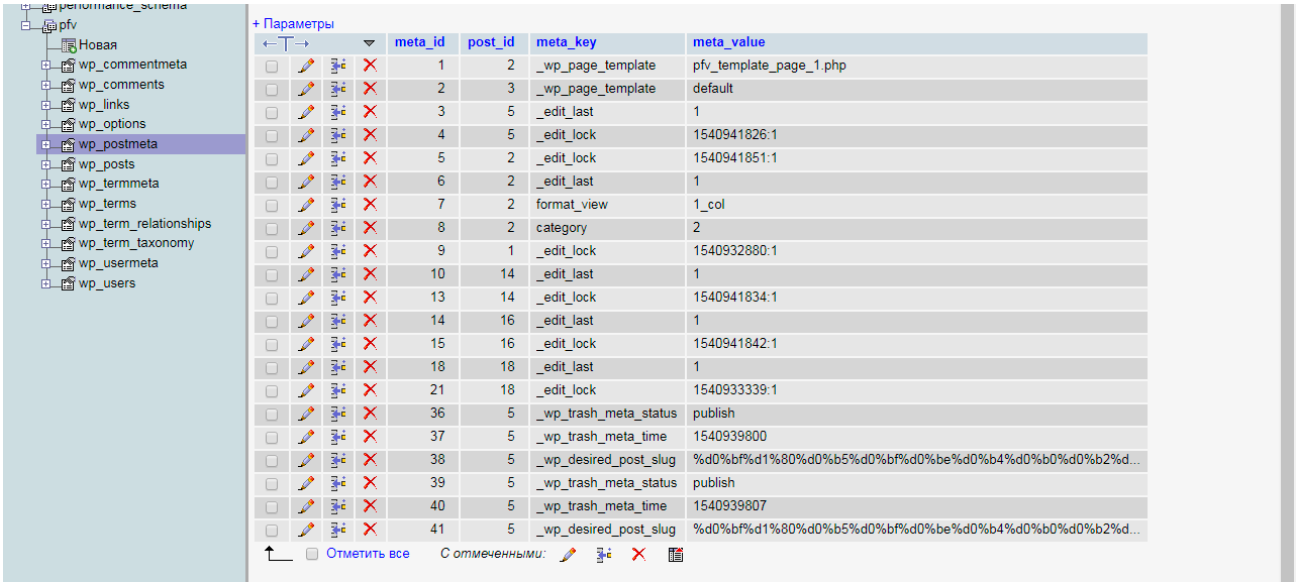


Рисунок 6.6 – Таблиця wp_postmeta

Таблиця зберігає в собі мета-дані з усього WordPress, а також плагіна. Оскільки плагін створює власні поля, то цю таблицю легко використовувати для їх зберігання. В результаті нові поля з таблиці використовуються для відображення додаткового контенту клієнтської частини. Відправляючи запити до wp_postmeta вдається отримувати потрібні шаблони для зміни формату виведення контенту.

В плагіні вся робота з базою даних здійснюється через властивості і методи глобальної змінної post. Завдяки цьому можна додавати, видаляти і редагувати записи в таблицях.

Основна робота проводиться над зв'язкою таблиць wp_posts і wp_postmeta. Саме вони необхідні для створення групи полів, потім створення конкретних полів і виведення контенту на цільових сторінках.

Для взаємодії з таблицями бази даних використовуються функції get_posts() и get_post_meta().

get_posts() – отримує записи (пости, сторінки, вкладення) з бази даних за зазначеними критеріями. Можна вибрати будь-які особливості і впорядкувати їх як завгодно. У get_posts() є встановлені параметри, які потрібно змінити, щоб вони не заважали отримати потрібний результат, наприклад:

- post_type – якщо ми вказуємо висновок таксономії для типу записів відмінний від post, то параметр post_type потрібно змінити, тому що він за замовчуванням завжди дорівнює post;
- numberposts – якщо ми забудемо змінити параметр numberposts, то довго можемо дивуватися чому виводиться тільки 5 записів замість потрібних 20.
- get_post_meta() – повертає значення зазначеного довільного поля запису (поста). Можна отримати масив всіх полів запису.

Для того, щоб отримати значення всіх довільних полів певного поста, потрібно залишити порожнім параметр $key.

WP_Query – це PHP клас, який дозволяє отримувати пости з бази даних за самими різними критеріями. Наприклад, ми можемо отримати пости:

- за певний проміжок часу;
- із зазначеної категорії, мітки;
- свіжі пости, випадкові пости, популярні пости;
- пости з зазначеними довільними полями або набором таких полів.

WP_Query використовується в WordPress завжди. Наприклад, під час генерації будь-якої сторінки WP_Query створює глобальну змінну $wp_query де зберігається інформація про поточні запити. На основі цієї інформації WordPress визначає на якій сторінці ми зараз знаходимося (пост, архів, мітка і т.д.). Також при побудові базового циклу WordPress, дані беруться зі змінної $wp_query і виводяться на екран через допоміжні функції: the_permalink(), the_content(). Також, $wp_query зберігає і інші дані. На різних типах сторінок дані різні. Подивитися які дані знаходяться в змінній $wp_query можна так:

```
global $ wp_query;
print_r ($ wp_query);
```

Часто отримати інформацію можна не працюючи безпосередньо з цим класом і глобальними змінними. Відбувається це завдяки набору функцій: умовні теги, теги шаблону відносяться до висновку даних всередині циклу і ряду інших функцій.

Для того, щоб взаємодіяти з базою данних при розробці плагіна також використовуються різні властивості функції WP_Query. Це дає можливість додавати, редагувати і видаляти записи з таблиць. При цьому основна перевага полягає в тому, що немає необхідності створювати контролер для роботи з базою даних. Всі маніпуляції з нею відбуваються в різних контролеррах в тому місці де це необхідно. Таким чином, вдалося спростити роботу з базою і розробити її відповідно до стандартів WordPress.

7 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ ПРОЕКТУ

7.1 Паттерн MVC

MVC – це не шаблон проекту, це конструкційний шаблон, який описує спосіб побудови структури нашого застосування, сфери відповідальності та взаємодію кожної з частин в даній структурі.

Вперше він був описаний в 1979 році, звичайно ж, для іншого оточення. Тоді не існувало концепції веб додатку. Тім Бернерс Лі (Tim Berners Lee) посіяв насіння World Wide Web (WWW) на початку дев'яностих і назавжди змінив світ. Шаблон, який ми використовуємо сьогодні, є адаптацією оригінального шаблону до веб розробки. Ідея, яка лежить в основі конструкційного шаблону MVC, дуже проста: потрібно чітко розділяти відповідальність для різного функціоналу в наших програмах:

Рисунок 7.1 – Загальна схема взаємодії компонентів MVC

Додаток розділяється на три основні компоненти, кожен з яких відповідає за різні завдання. Контролер керує запитами користувача (одержувані у вигляді запитів HTTP GET або POST, коли користувач натискає на елементи інтерфейсу для виконання різних дій). Його основна функція – викликати і координувати дію необхідних ресурсів і об'єктів, потрібних для виконання дій, що задаються користувачем. Зазвичай контролер викликає відповідну модель для задачі і вибирає відповідний вид.

Модель – це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління додатком. У будь-якому додатку вся структура моделюється як дані, які обробляються певним чином. Тільки дані, які повинні бути оброблені відповідно до правил (дата не може вказувати в майбутнє, e-mail повинен бути в певному форматі, ім'я не може бути довшим х символів, і так далі). Модель дає контролеру уявлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти їх користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних. Модель містить найбільш важливу частину логіки нашого застосування, логіки, яка вирішує завдання, з якою ми маємо справу (форум, магазин, банк, тощо). Контролер містить в основному організаційну логіку для самого додатка (дуже схоже на ведення домашнього господарства).

Вид забезпечує різні способи представлення даних, які отримані з моделі. Він може бути шаблоном, який заповнюється даними. Може бути кілька різних видів уявлень, і контролер вибирає, який підходить найкраще для поточної ситуації. Веб додаток зазвичай складається з набору контролерів, моделей і уявлень. Контролер може бути влаштований як основний, який отримує всі запити і викликає інші контролери для виконання дій в залежності від ситуації.

Очевидною перевагою в процесі розробки плагіну, яку ми отримуємо від використання концепції MVC – це чіткий поділ логіки уявлення (інтерфейсу користувача) і логіки програми. Підтримка різних типів користувачів, які

використовують різні типи пристроїв є спільною проблемою наших днів. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних. У даній ситуації це допомогло зробити систему адаптивною під різні пристрої і в залежності від того з чого користувач заходить на сайт контролер поверне йому відповідний для використання варіант дизайну. Крім ізолювання видів від логіки додатку, концепція MVC істотно зменшує складність великих додатків. Код виходить набагато більш структурованим, і, тим самим, полегшується підтримка, тестування і повторне використання рішень.

Поділ відповідальності і гнучкість дозволили налаштовувати різні компоненти платформи на свій розсуд. Змінювати будь-яку частину конвеєра роботи MVC або адаптувати його до своїх потреб. Відповідно, на відміну від таких стандартних засобів для проектування плагінів, які промонуэ CMS Wordpress, патерн проектування дав можливість реалізувати розширення з усіма задуманими спочатку можливостями. У даній ситуації інформація про такі сутності, як вид полів, дії над ними, шаблони для динамічного виводу даних та інше стало розміщуватися всередині моделей. Взаємодію між ними, разом з описом логіки, вдалося розмістити в контролерах. За відображення інформації на сторінках відповідають уявлення.

На самому початку варто визначити, що плагін розробляється з використанням мови PHP. Відповідно, треба сказати про особливості паттерна MVC в рамках цієї мови.

Особливістю при використанні MVC в PHP, є те, що існує одна точка входу в PHP додаток, яка, наприклад, досягається наступним чином. Створюється index.php через який будуть оброблятися всі запити, для цього створюємо в папці з індексом файл .htaccess і поміщаємо в нього такий код:

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?route=$1 [L,QSA]
```

У наданому коді, першим рядком, перевіряється існування запитуваного файлу, і якщо його немає, то йде перенаправлення на index.php, інакше навіть запити картинок сайту будуть перенаправлятися на index. Останній рядок коду перетворює запити виду index.php?route=chat/index у вид index.php/chat/index. Якщо у вас немає можливості використовувати ModRewrite в своєму додатку, то вам доведеться робити переадресацію вручну.

Варто відзначити, що всі сайти робляться за таким принципом, в якому точкою для входу є файл index.php. Саме в ньому знаходиться основний шаблон або шаблон головної сторінки, це Landing Page, який складається всього з однієї сторінки. У випадки з плагіном index.php також присутній, він грає роль головного файлу, до якого браузер звертається при активації плагіна в WordPress. При цьому створювати всередині файл .htaccess не потрібно, тому що він і так присутній в кореневій директорії сайту і всередині знаходяться потрібні налаштування. Також варто відзначити, що роблячи плагін, використовуючи правильну структуру, WordPress сам зрозуміє до якого файлу звертатися, щоб розширення запрацювало.

PHP модель. Дані про PHP моделі містяться в їх атрибутах і можуть бути змінені тільки через спеціальні функції. Модель може містити в собі кілька подань. Як правило, php модель це клас який працює з бази даних, конкретніше: запис, читання, видалення. Природно читання інформації з бази даних може бути реалізовано кількома методами (функціями). Як приклад модель статей на сайті: можна отримати конкретну статтю з бази даних, список останніх, популярних, якоїсь категорії – це все налаштування моделі. Для наочності нижче надано приклад PHP моделі:

```
<?php
    function methodName()
      {
      $link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
      if (!$link) {
      die('Could not connect: ' . mysql_error());
```

```
        }
        echo ‘Connected successfully’;
        mysql_close($link);
        $query_results= mysql_query(’select * from searchNames order by firstname
  desc’);

        $data = array();
        while ($row = mysql_fetch_objects($query_results)) {
        $data[] = $row;
        }
        return $data;
        }
?>
```

PHP контролер. Контролер отримує запити користувачів, які ми направляємо через index.php, і відповідно до них, коректує роботу моделі. Правильніше сказати контролює роботу PHP додатку. Приклад:

```
<?php
        $data= methodName();
        display_template(’data.tpl’);
?>
```

PHP уявлення. Ця сутність відстежує зміну в моделі і створює або змінює інтерфейс PHP додатку. По суті уявлення выводить на екран те що хотів відобразити розробник. Наприклад:

```
<html>
<body>
        <h1>List of Datas</h1>
        <?php foreach ($data as $row) { ?>
        <h2><?php echo $row->firstname ?></h2>
```

```
        <h2><?php echo $row->lastname?></h2>
        <?php } ?>
</body>
</html>
```

Працює шаблон MVC в даній ситуації наступним чином. При зверненні користувача за потрібною url-адресою вибирається відповідний контролер, який звертається до подання і моделі, і виводиться інформація. Іншими словами контролер в MVC є сполучною ланкою моделі та подання.

Переваги у такого підходу при програмуванні на PHP складно переоцінити. Як згадувалося вище, це, перш за все, диференціація розробки PHP сайту на відділи. Також збільшується швидкість роботи PHP додатку, якщо створюється великий проект. Ну і те, що стосується безпосередньо самого PHP-розробника, це правильна структуризація коду (все на своїх місцях, так легше для розуміння).

Весь функціонал паттерна MVC у плагіні реалізується з використанням наступної структури проекту:

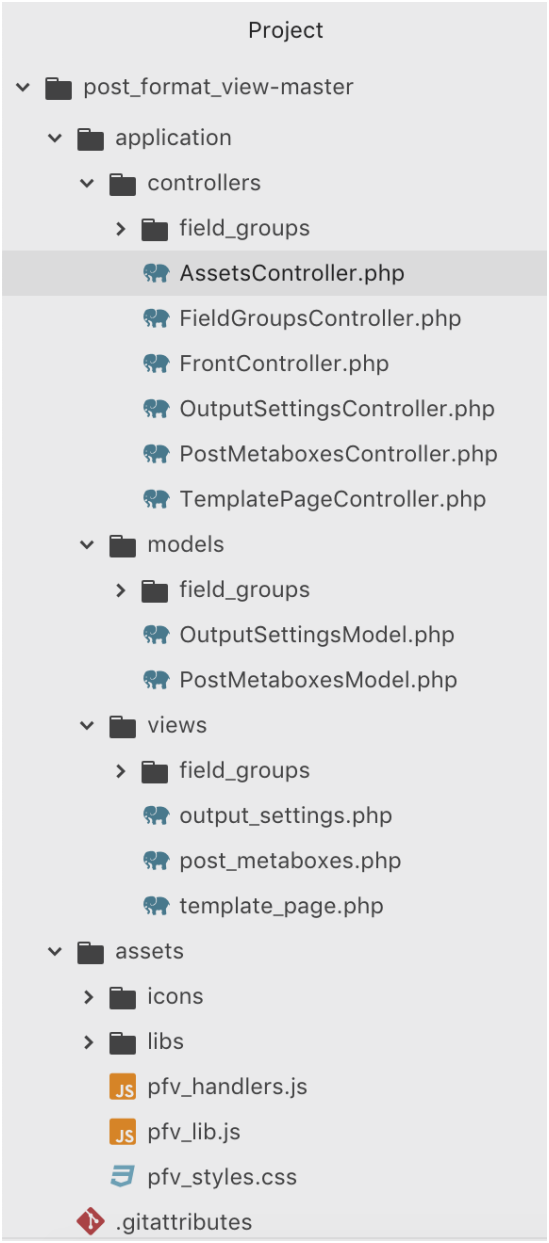


Рисунок 7.2 – Файлова структура плагіну

У корені плагіна лежить папка application, в якій всі основні файли, папка assets, де знаходяться допоміжні бібліотеки для роботи і файл post_format_view.php – це точка входу для плагіна, в якій підключений основний контролер. Код виглядає так:

```php
<?php
/**
 * Plugin Name: post_format_view
 * Description: Описание плагина желательно не очень длинное
 * Plugin URI:  Ссылка на инфо о плагине
 * Author URI:  Ссылка на автора
 * Author:      Имя автора
 * Version:     Версия плагина, например 1.0
 *
 * Text Domain: Идентификатор перевода, указывается в load_plugin_textdomain()
 * Domain Path: Путь до файла перевода. Нужен если файл перевода находится не в той же папке, в которой находится теку
 *              Например, .mo файл находится в папке myplugin/languages, а файл плагина в myplugin/myplugin.php, тогда
 *
 * License:     GPL2
 * License URI: https://www.gnu.org/licenses/gpl-2.0.html
 *
 * Network:     Укажите "true" для возможности активировать плагин по все сети сайтов (для Мультисайтовой сборки).
 */


require_once('application/controllers/FrontController.php');

$front = new FrontController();
```

Рисунок 7.3 – Точка входу в плагін post_format_view.php

Спочатку йде довгий коментар, який необхідний для розпізнавання плагіна по стандартам WordPress. Розробник повинен надати базовий опис свого розширення, заповнивши поля назви, опис, посилання на інформацію про плагін, посилання на автора, номер версії та багато іншого.

Далі йде підключення головного контролера FrontController.php і потім його виклик за допомогою створення нового об'єкту класу.

Тепер варто розглянути вміст папок, щоб чітко розуміти який файл за що відповідає.

В папці assets лежать тільки допоміжні бібліотеки та стилі:

- pfv_lib.js – файл зі скриптами, які необхідні для роботи;
- pfv_handlers.js – файл з обробником подій для функцій з pvf_lib.js;
- pvf_style.css – загальний файл всіх стилів для оформлення зовнішнього вигляду плагіна в панелі адміністратора.

Також є папка icons, де розташовані іконки, щоб зробити інтерфейс плагіна більш практичним. Крім неї присутня папка libs, де знаходяться готові бібліотеки для роботи.

У даній розробці вони будуть використовуватися по мінімуму і лише для спрощення реалізації деяких елементів інтефейсу.

Основою плагіна є папка application, де розміщені контороллери, моделі та уявлення (всі вони згруповані по папкам для зручності). Почати варто з того за що відповідають представлені контролери:

- FrontController.php – основний контролер, в якому зібрані підключення всіх інших;
- AssetsController.php – підключення різних допоміжних матеріалів для роботи (скрипти, стилі і т.д.);
- FieldGroupsController.php – підключає модулі, які відповідальні за роботу групи полів;
- OutputSettingsController.php – підключає вивід метабоксів на кожну сторінку редагування для налаштувань формату виведення контенту. У метабоксі також вказується з якої рубрики брати записи і формат представлення (шаблон верстки для відображення);
- PostMetaboxesController.php – вивід групи полів на цільовій сторінці, щоб потім заповнювати і редагувати окремі поля;
- TemplatePageController.php – створення шаблону і його розміщення в потрібному місці.

Також серед контролерів присутня папка field_group, в якій згруповані контролери для роботи виключно з групою полів:

- FG_MetaboxController.php – створення, редагування і видалення полів в конкретній групі полів;
- FG_PostTypeController.php – робота з группою полів в базі даних і створення нових груп полів;
- FieldPostTypeController.php – прив'язка поля до конкретної групи полів, вибір типу одного поля, робота з полями в базі даних;

- LocationsMetaboxController.php – відповідає за розташування групи полів, створення місце, де вони перебуватимуть, тобто підбір конкретної рубрики.

Це перелік всіх контролерів, які беруть участь в розробці. Під них були створені спеціальні моделі, але тільки в тих ситуаціях, де вони дійсно потрібні для роботи з даними. Серед моделей, які є можна виділити:

- OutputSettingsController.php – модель яка відповідає за вид формату виведення контенту і в яку рубрику його додавати;
- PostMetaboxesController.php – налаштування для виведення групи полів на конкретній сторінці, щоб в подальшому можна було наповнювати їх котентом і редагувати.

Також, щоб структурувати моделі, деякі з них були додані в папку field_groups. За рахунок цього стає зрозуміло, що дані моделі відповідають за роботу саме з полями. У проекті представлені наступні моделі для роботи з полями:

- FG_MetaboxController.php – відповідає за додавання полів, їх редагування і видалення;
- FG_PostTypeController.php – створює групи полів і працює з ними в базі даних;
- FieldPostTypeController.php – працює з конкретними полями, додає їх в групи і дає можливість вибрати конкретний тип;
- LocationsMetaboxController.php – допомагає створювати конкретне розташування для групи полів, щоб надалі з ними можна було працювати на цільовій сторінці.

Останньою складовою патерна MVC є уявлення. Всі вони зібрані в папці views та відповідають за відображення конкретних елементів інтерфейсу плагіна:

- output_settings.php – сторінка, на якій користувач вибирає формат виведення і рубрику, куди поміщати групу полів;
- post_metaboxes.php – інтерфейс додавання полів в групу полів, відображення всього цього в панелі адміністратора;

– template_page.php – базовий шаблон для сторінки налаштувань плагіна, перед формуванням групи полів і її виведенням на цільовій сторінці.

Крім цього є ще й уявлення в папці field_groups, які відповідають за користувальницький інтерфейс роботи з конкретними полями:

– fg_metabox.php – призначений для користувача інтерфейс створення конкретного поля;
– locations_metabox.php – інтерфейс для вибору місця розташування групи полів за допомогою параметрів.

Таким чином структура плагіна одночасно відповідає і стандартам WordPress, і принципам побудови патерну MVC. Ключова перевага полягає в можливості легко розширювати плагін. При необхідності можна створити нову модель або контролер і все це зв'язати. Підключення моделі буде відбуватися в новому контролері, а він вже без проблем з'єднається з головним FrontController і результат його роботи буде виведений в потрібне подання. Якщо ж в процесі додавання нових функцій знадобиться розробити тільки контролер, то все буде ще простіше. Досить просто підключити його до головного контролера і далі результат роботи функцій всередині нового контролера вийде викликати в будь-якому поданні. Діаграмма класів виглядає наступним чином:

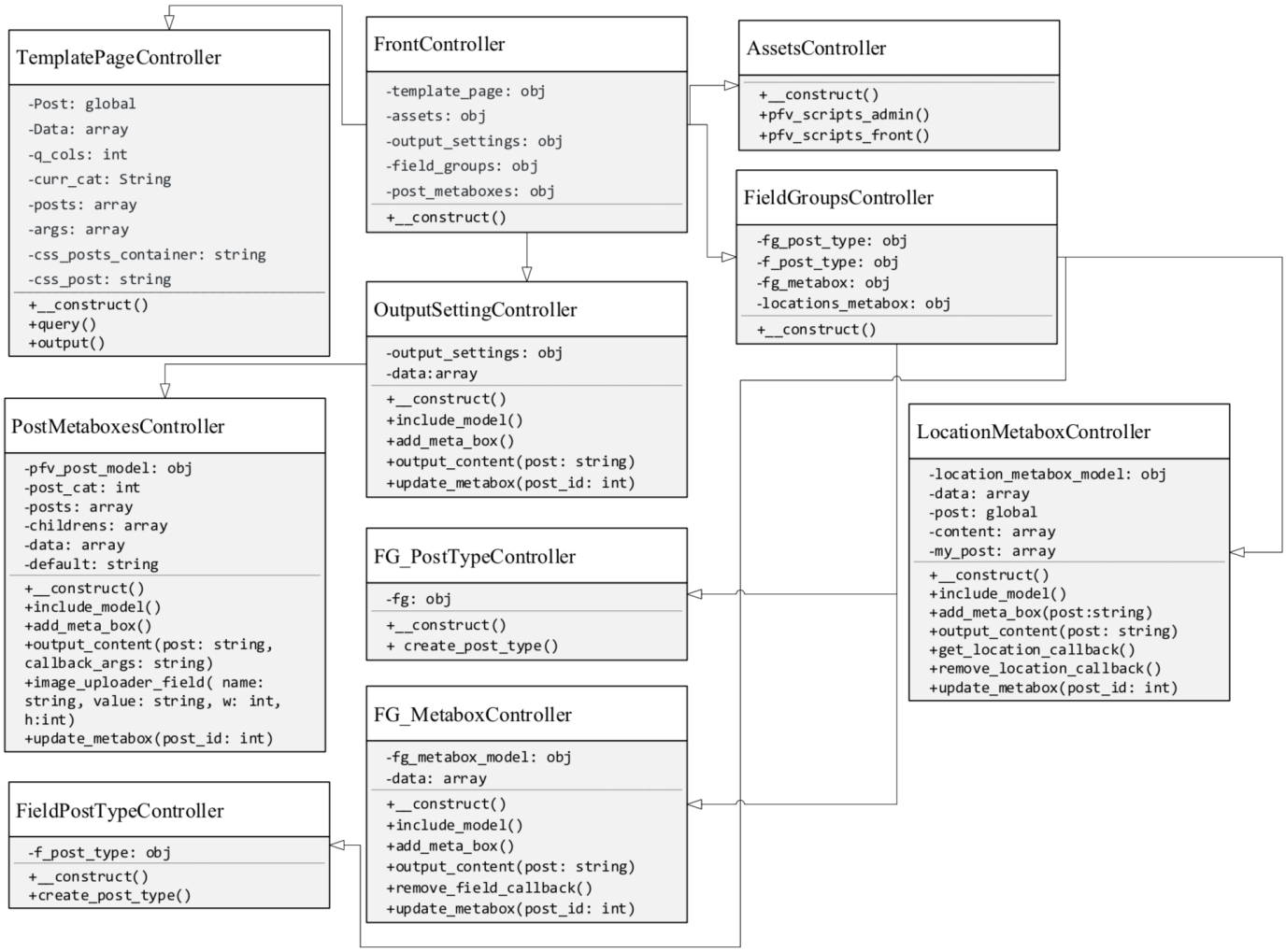


Рисунок 7.4 – Діаграмма класів

7.2 Контроллери

Щоб зрозуміти з чим працює плагін і як йому вдається створювати контент, редагувати його і потім виводити в потрібному форматі необхідно розібратися в роботі контролерів. Саме в них прописаний код, який реалізує функціонал плагіна.

Будь-який сайт на WordPress після завантаження, автоматично створює масив з активними плагінами. Ядро системи управління вмістом робить це автоматично і проходиться по всіх елементах масиву в пошуках активних плагінів. Відповідно, з цього і починається робота плагіна, який представлений в магістерській дисертації.

Елемент масиву передає управління файлу post_format_view.php, який є точкою входу для початку роботи плагіна.

У ньому міститься підключення основного контролера програми FrontController.php і відразу ж створює екземпляр класу FrontController. На цьому етапі і починається робота плагіна. Код основного контролера виглядає так:

```php
<?php

class FrontController{

	function __construct(){

		require_once('TemplatePageController.php');

		require_once('AssetsController.php');

		require_once('OutputSettingsController.php');

		require_once('FieldGroupsController.php');

		require_once('PostMetaboxesController.php');

		$template_page = new TemplatePageController();

		$assets = new AssetsController();

		$output_settings = new OutputSettingsController();

		$field_groups = new FieldGroupsController();

		$post_metaboxes = new PostMetaboxesController();

	}
}
```

Рисунок 7.5 – Контроллер FrontController.php

У FrontController відбувається підключення всіх інших контролерів для роботи плагіна, а також створюються екземпляри їх класів для активації роботи контролера.

Першим в списку йде TemplatePageController. Цей контролер є основним для роботи з форматами виведення котента.

Оскільки ця функція має на увазі зміну візуальної частини інформації, яка прописується в групі полів, то для будь-якого виду нам потрібна окрема верстка. В даному випадки шаблон з нею, який буде створюватися, коли користувач в панелі адміністратора вибере вид формату виведення. Саме такий підхід дозволив створити кілька форматів і розширити надалі плагін, щоб варіантів стало ще більше. Контролер відповідає за те, щоб шаблон створювався автоматично. Тобто користувачеві досить вибрати варіант в панелі адміністратора WordPress і в самій структурі файлів з'явиться верстка вже з прописаними полями з групи. У браузері інформація відобразитися по вибранному шаблону, код який відповідає за це виглядає так:

```php
<?php

class TemplatePageController {

	function __construct() {

		if (file_exists(WP_PLUGIN_DIR.'/post_format_view/application/views/template_page.php')){
		//Содержимое, которое будем записывать в файл
		$content = '<?php /* Template Name: PFV_template_page */ include_once(WP_PLUGIN_DIR . \'/post_format_view/application,
		}
		//Проверяем существование файла
		if (!file_exists(get_template_directory().'/pfv_template_page.php')){
		// Если файл не существует, то создаем его
		$fp = fopen(get_template_directory().'/pfv_template_page_1.php', 'w');
		// записываем в файл содержимое
		fwrite($fp, $content);
		// закрываем
		fclose($fp);
		}

	}

}
```

Рисунок 7.6 – Контроллер TemplatePageController.php

Працює це таким чином. Контролер перевіряє чи є вже створений шаблон для виведення контенту. Якщо він є, то записує в файл вміст, який користувач вказав в панелі адміністратора.

У тому випадку якщо раніше був створений шаблон і користувач хоче зробити новий, то йде чергова перевірка. Якщо вже є шаблон, то новий буде створюватися з

іншим ім'ям і таким чином, плагін дає можливість для кожної сторінки в WordPress мати по кілька шаблонів для виведення інформації.

Оскільки цей контролер відповідає тільки за створення самих шаблонів, то він йде першим, а вже далі в роботу вступають інші елементи управління всередині плагіна.

Далі в основному контролері підключається AssetsController. Його основне завдання полягає в тому, щоб підключити сторонні бібліотеки і скрипти.

Наприклад, на javascript спеціально був розроблений простий слайдер, щоб користувач міг скористатися таким форматом виведення даних на своєму сайті. Брати готові рішення було недоцільно, тому що їх багато і після впровадження конкретної версії одного слайдера, він може конфліктувати з існуючими. Окремий скрипт для цього всього спеціально під плагін гарантує, що формат виведення у вигляді слайдера точно буде працювати справно.

Також підключаються такі бібліотеки, як fancybox виключно для того, щоб зробити для користувача інтерфейс більш красивим і інтуїтивно зрозумілим. Крім цього AssetsController включає і підключення інших файлів, наприклад, стилів для сторінки налаштування плагіна і елементів управління їм.

Важливою частиною плагіна є OutputSettingsController. Він відповідає за створення метабоксів. Це блоки з полями, які в подальшому можуть зберігатися в базі даних. Потім з їх допомогою можна працювати з базою даних і отримувати з неї інформацію.

Контролер містить функції для управління метабоксами. Вони в свою чергу містять список існуючих категорій на сайті, які раніше створював користувач через панель адміністратора і список варіантів виведення (абзац, кілька колонок, слайдер і т.д.). Код виглядає так:

```php
<?php

class OutputSettingsController{

	function __construct(){

		add_action( 'add_meta_boxes', array($this, 'add_meta_box') );

		add_action('save_post', array($this, 'update_metabox'), 0);
	}

	function include_model(){

		require_once(WP_PLUGIN_DIR.'/post_format_view/application/models/OutputSettingsModel.php');

		$output_settings = new OutputSettingsModel();

		return $output_settings;
	}

	function add_meta_box(){
		add_meta_box('pfv_output_settings','Настройки вывода записей',array($this, 'output_content'),'page', 'normal',
	}

	function output_content($post){

		$data = $this->include_model()->output_data($post);

		require_once(WP_PLUGIN_DIR.'/post_format_view/application/views/output_settings.php');
	}

	function update_metabox($post_id){

		$this->include_model()->update_metabox($post_id);

		return $post_id;
	}

}
```

Рисунок 7.7 – Контроллер OutputSettingsController.php

Коли користувач працює з плагіном в панелі адміністратора WordPress, він вибирає цільову сторінку, а потім на ній потрібний йому формат виведення. Після цього дані на сторінці зберігаються і в цей момент OutputSettingsController зберігає їх в базі даних. Попутно контролер TemplatePageController створює шаблон для виведення, який теж зберігається. В майбутньому це дасть можливість зробити копію сторінки і вибрати для неї вже готовий шаблон з раніше заданим форматом виведення.

По суті першою функцією плагіна є вибір формату виведення. Як мінімум, якщо поставити на свій сайт на WordPress даний плагін і не використовувати довільні поля, то він дозволить змінити формат виведення контенту на сторінках. За це відповідають

вишеописані контролери. Всі інші додають в розширення другу важливу функцію, а саме створення групи полів для додавання і редагування контенту на сторінках готового сайту.

Для роботи довільних полів в плагіні виділена ціла директорія контролерів під назвою field_groups. У ній зберігаються всі контролери для управління довільними полями. Їх підключення відбувається в FieldGroupsController, який виділений, як основний для цієї функції плагіна:

```php
<?php

class FieldGroupsController{

	function __construct(){

		require_once('field_groups/FG_PostTypeController.php');

		require_once('field_groups/FieldPostTypeController.php');

		require_once('field_groups/FG_MetaboxController.php');

		require_once('field_groups/LocationsMetaboxController.php');

		$fg_post_type      = new FG_PostTypeController();

		$f_post_type       = new FieldPostTypeController();

		$fg_metabox        = new FG_MetaboxController();

		$locations_metabox = new LocationsMetaboxController();

	}
}
```

Рисунок 7.8 – Контроллер FieldGroupsController.php

Контролер відповідає за підключення інших контролерів і створення примірників їх класів. Все це активує функцію плагіна по роботі з довільними полями.

Щоб зрозуміти роботу плагіна конкретно з групою полів варто розібрати кожен контролер з директорії filed_groups окремо.

Головним контороллем тут є FG_PostTypeController. Даний контролер відповідає за створення типу записа – «група полів». Він необхідний для створення повноцінної групи довільних полів і подальшого виведення їх на обрану користувачем цільову сторінку. Код виглядає так:

```php
<?php

class FG_PostTypeController{

	function __construct() {
		add_action('init', array($this,'create_post_type'));
	}

	function create_post_type(){
		require_once(WP_PLUGIN_DIR.'/post_format_view/application/models/field_groups/FG_PostTypeModel.php');
		$fg = new FG_PostTypeModel;
		$fg->create_post_type();
	}

}
```

Рисунок 7.9 – Контроллер FG_PostTypeController.php

Робота контролера FG_PostTypeController безпосередньо пов'язана з відповідною моделлю. Основне завдання цього програмного блоку плагіна полягає в тому, щоб створити екземплян класу, який і збереже в базу повноцінну групу полів. Також з його допомогою можна видаляти групи полів з панелі адміністратора і вибирати для них цільову сторінку, де буде виводиться група.

Далі йде FiledPostTypeController, який безпосередньо пов'язаний з попереднім контролером. Оскільки група полів являє собою структуру, яка містить в собі поля різного типу, то з кожним з них окремо також необхідно працювати.

Даний контролер відповідає за кожне окреме поле і потрібен для того, щоб працювати з їх даними. Він надає можливість створювати поля різних типів і видаляти їх.

У момент створення в базу даних записуються такі значення, як ярлик поля, машинне ім'я і тип (вибір різновиду поля – текстове, список, зображення і т.д.).

Кожне поле в плагіні має унікальний id батька, щоб можна було прив'язати їх до конкретної групи полів. У базі даних це значення зберігається в колонці таблиці під назвою post_parent.

Створення групи полів реалізовано з використанням метабоксів. Це спеціальний тип сутності в WordPress, який призначений для зберігання даних. У магістерській дисертації контролер FG_MetaboxController відповідає саме за їх створення.

Його основний обов'язок створення метабоксу для групи полів і додавання в нього полів за допомогою AJAX запитів, тобто динамічно, без оновлення сторінки. Контролер також займається і видаленням полів.

При зверненні до FG_MetaboxController відбувається створення нового запису в базі даних типу «поле». Також в цей запис додаються дані про конкретні поля, які створив користувач. Виглядає контролер так:

```php
<?php

class FG_MetaboxController{

	function __construct(){

		add_action( 'add_meta_boxes', array($this, 'add_meta_box') );

		add_action('wp_ajax_update_field', array($this, 'update_field_callback'));

		add_action('wp_ajax_remove_field', array($this, 'remove_field_callback'));

	}

	function include_model(){

		require_once(WP_PLUGIN_DIR.'/post_format_view/application/models/field_groups/FG_MetaboxModel.php');

		$fg_metabox_model = new FG_MetaboxModel();

		return $fg_metabox_model;
	}

	function add_meta_box(){
		add_meta_box('pfv_field_group','Группа полей',array($this, 'output_content'),'pfv_field_groups', 'normal',
	}

	function output_content($post){

		$data = $this->include_model()->output_data();

		require_once(WP_PLUGIN_DIR.'/post_format_view/application/views/field_groups/fg_metabox.php');
	}

	function remove_field_callback(){

		$this->include_model()->remove_field();

	}

	function update_field_callback(){

		$this->include_model()->update_field();
	}

}
```

Рисунок 7.10 – Контроллер FG_MetaboxController.php

Робота контролера починається з включення конструктора. У ньому відбувається обробка методів для додавання метабокса, додавання поля і видалення поля.

Далі відбувається робота з методом include_model, який відповідає за підключення відповідної моделі і створює екземпляр класу.

Після цього відбувається обробка подій, які були описані в методах вище, в контструкторі. Також є метод output_content, який займається виводом вмісту метабоксу.

Оскільки WordPress зроблений за допомогою процедурного програмування, то більшість дій можна робити за допомогою готових функцій. У нашому випадку робота контролера FG_MetaboxController відбувається саме за такою схемою, що дає можливість реалізувати конкретні функції плагіна шляхом написання необхідних методів.

Останній контролер в директорії field_groups – це LocationsMetaboxController. Він відповідає за створення налаштування місця виведення групи полів. Робиться це, як і в випадку з полями через метабокси. Оскільки після створення групи полів користувачеві необхідно відобразити їх на конкретній сторінки в панелі адміністратора даний контролер необхідний.

У WordPress є функціонал, який дозволяє створювати рубрики для контенту. Плагін допомагає розмістити групу полів на тих сторінках або записах, які відносяться до конкретної рубрики. Тобто відразу потрібно створити рубрику, що є стандартною опцією WordPress, потім групу полів, заповнити її необхідними полями і потім вже працювати з конктентом на сторінці.

Даний контролер створює місце розташування для групи полів, дозволяє його редагувати і видалити якщо в полях вже немає необхідності на конкретній сторінці. Виглядає контролер так:

```php
class LocationsMetaboxController{

	function __construct(){
		add_action( 'add_meta_boxes', array($this, 'add_meta_box') );

		add_action('wp_ajax_get_location', array($this, 'get_location_callba

		add_action('wp_ajax_remove_location', array($this, 'remove_location_

		add_action('save_post', array($this, 'update_metabox'));
	}

	function include_model(){

		require_once(WP_PLUGIN_DIR.'/post_format_view/application/models/fie

		$location_metabox_model = new LocationsMetaboxModel();

		return $location_metabox_model;
	}

	function add_meta_box($post){
		add_meta_box('pfv_location_settings','Настройки месторасположения вы
			'pfv_field_groups', 'normal', 'low');
	}

	function output_content($post){

		$data = $this->include_model()->output_data($post);

		require_once(WP_PLUGIN_DIR.'/post_format_view/application/views/fiel
	}

	function get_location_callback(){

		$this->include_model()->get_location();
	}

	function remove_location_callback(){

		$this->include_model()->remove_location();
	}

	function update_metabox($post_id){

		//if ( ! wp_verify_nonce($_POST['extra_fields_nonce'], __FILE__) ) r
		if ( defined('DOING_AUTOSAVE') && DOING_AUTOSAVE  ) return false; //
		if ( !current_user_can('edit_post', $post_id) ) return false; // вых
		if( !isset($_POST['location_settings']) ) return false; // выходим е

		$content = array();
		$_POST['location_settings'] = array_map('trim', $_POST['location_set
		foreach( $_POST['location_settings'] as $key => $value ){
			$content['locations'][] = array('name' => $key, 'value' => $v
		}

		// Создаем массив данных
		$my_post = array();
		$my_post['ID'] = $post_id;
		$my_post['post_content'] = serialize($content);

		if ( ! wp_is_post_revision( $post_id ) ){
				// удаляем этот хук, чтобы он не создавал бесконечно
				remove_action('save_post', array($this, 'update_metab

				// обновляем пост, когда снова вызовется хук save_po
				wp_update_post($my_post);

			}

		return $post_id;
```

Рисунок 7.11 – Контроллер LocationsMetaboxController.php

Коли користувач вибирає розташування для групи полів, контролер записує дане значення в базу даних в поле post_content в таблиці.

Далі вже на конкретній сторінці в панелі адміністратора видно поля для додавання контенту, редагування і також їх можна видалити при необхідності.

Рухаючись далі по роботі плагіна переходимо до контролера PostMetaboxesController. Описані вище контролери відповідали за те, щоб створити групу полів, додати в неї поля конкретного типу і вибрати цільову сторінку для відображення. Тепер необхідно, якось відображати групи полів на сторінках в панелі адміністратора, щоб користувач міг задати формат виводу.

Контролер PostMetaboxesController потрібен для того, щоб створити метабокс, куди запишеться створена користувачем група полів.

Основний метод контролера add_meta_box виглядає так:

```
function add_meta_box(){

        global $post;
        $post_cat = array();
        $post_cat = wp_get_post_categories($post->ID);
        $posts = get_posts(array('numberposts' => -1,'post_type'=>'pfv_field_groups'));
        $childrens = array();

        foreach ($posts as $_post) {

                $locations = unserialize($_post->post_content)['locations'];
                if(is_array($locations)){
                        foreach ($locations as $location) {
                                if(in_array($location['value'], $post_cat)){//если значение location в группе полей совпадает с масс

                                $childrens = get_children(array('post_type'=>'pfv_field', 'post_parent'=> $_post->ID));
                                add_meta_box('pfv_post',$_post->post_title,array($this, 'output_content'),'post', 'normal', 'high',a

                                }
                        }
                }
        }
}
```

Рисунок 7.12 – Метод контроллера PostMetaboxesController.php

Метод add_meta_box спочатку дізнається до яких рубриках відноситься поточний запис. Після цього він перебирає всі записи типу «група полів» в базі даних, щоб знайти потрібну. Таким чином він отримує значення locations у підходящому користувачеві записі і в поле post_content записує його.

Далі відбувається порівняння. Необхідно зіставити масив зі значеннями існуючий рубрик і тим, в якому зберігаються групи полів. Якщо збіг є, то метод бере дочірні записи групи полів, а це в нашому випадку звичайні поля, записує їх в масив і передає в уявлення. Таким чином, користувач на цільовій сторінці бачить групу полів і може з нею працювати.

На цьому опис роботи контролерів плагіна завершено. Він надає користувачеві відразу дві можливо, а саме вибір формату виведення контенту і створення групи полів для того, щоб додавати в неї контент і редагувати його в подальшому. Обидва механізми працюють самостійно і користувач може використовувати, як один з них, так і обидва відразу. Звичайно, створення власної групи полів і зміна формату виведення для кожної сторінки виявиться більш ефективним способом роботи з контентом на сайті і його редагуванням.

7.3 Моделі існуючих сутностей

Модель – це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління додатком. У будь-якому додатку вся структура моделюється як дані, які обробляються певним чином. Що таке користувач для додатка – повідомлення або книга? Тільки дані, які повинні бути оброблені відповідно до правил (дата не може вказувати в майбутнє, e-mail повинен бути в певному форматі, ім'я не може бути довшим X символів, і так далі).

Модель дає контролеру уявлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти її користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних.

Модель містить найбільш важливу частину логіки нашого застосування, логіки, яка вирішує завдання, з яким ми маємо справу (форум, магазин, банк, тощо). Контролер містить в основному організаційну логіку для самого додатка (дуже схоже на ведення щоденика справ на кожен день).

Оскільки в плагіні необхідно працювати з декількома типами даних, то для деяких з них створювалися окремі моделі. Тільки за допомогою чітко описаних правил в моделях можна досягти правильного регультата роботи.

Важливою складовою плагіна є модель OutputSettingsModel. У ній описані правила для метабоксів, які необхідні, щоб зберігати формати виведення. Модель пов'язана з контролером OutputSettingsController, який записує дані по частині створення метабоксів в таблицю бази даних. Також він обробляє правила з моделі для того, щоб вибрати цільову сторінку.

Основний метод моделі output_data призначений для зберігання масивів з даними. Тут здійснюються різноманітні вибірки і запис результатів в змінні. Щоб контролер в подальшому працював корретно необхідно, щоб в моделі були описані правила, які збирають усі формати висновків, категорії з бази даних і описані формати виведення.

Також в моделі міститися правила для опису процедури поновлення метабоксів. За допомогою методу update_metabox перевіряється наявність інформації в полях, її збереження і видалення.

Якщо користувач захоче щось змінити, то після натискання кнопки в панелі адміністратора «Опублікувати» зміни вступлять в силу. Модель автоматично робить оновлення та збереження даних.

Також у нас є модель PostMetaboxesModel, яка відповідає за створення метабоксів, де розташовується інформація про місце виведення групи полів. Коли модель обробить розписані правила, їх можна передавати контролеру PostMetaboxesController для подальшої роботи і виведення інформації в уявлення.

Основним методом моделі є update_metabox, який займається оновленням інформації в метабоксі. При цьому його ключова задача полягає в тому, щоб проводити перелік перевірок. Тільки зареєстровані користувачі можуть змінювати групу полів або видаляти з неї поля. Також потрібно виключити автоматичне збереження після змін. Користувач повинен самостійно зберігати зміни, натискаючи відповідну кнопку в панелі адміністратора.

Модель ще і проходить по всім полям у масиві, щоб знайти порожні і автоматично видалити їх. Якщо в полях немає інформації і вони не використовуються, то немає ніякого сенсу залишаючи їх в групі полів.

Далі у нас є цілий перелік моделей для роботи з групою полів. В принципі так само в магістрескій роботі відокремлені і контролери, які відповідають за цю функцію.

В папці field_groups міститься відразу 4 моделі, які відповідають за певні дії плагіна. Почнемо з моделі FG_MetaboxModel. Саме в ній описані правила для створення групи полів за допомогою метабоксів і створення окремих полів з певним типом.

Також модель займається оновленням полів і зверненням до бази даних, щоб редагування пройшло успішно. Через неї відбувається і видалення полів у кожній конкретній групі.

Не менш важливою моделлю є і LocationsMetaboxModel, завдання якої створити місце для розміщення групи полів на конкретній сторінці, змінити його якщо буде потрібно і видалити. Всі ці процедури описані в даній моделі.

Працює вона з базою даних і бере з неї значення рубрик, а також місця розташування записів. Потім присвоює певній групі полів необхідне розташування. Користувач вибирає це в панелі адміністратора, а модель робить так, щоб контролер коректно вивів групу, зберіг її, дав можливість редагувати місце виведення і навіть видаляти якщо поля більше не потрібні.

Основний метод моделі output_data призначений для зберігання масивів з даними. Тут здійснюються різноманітні вибірки і запис результатів в змінні. Щоб контролер в подальшому працював корретно необхідно, щоб в моделі були описані правила, які збирають усі формати висновків, категорії з бази даних і описують формати виведення.

Також в моделі міститися правила для опису процедури поновлення метабоксів. За допомогою методу update_metabox перевіряється наявність інформації в полях, її збереження і видалення.

Якщо користувач захоче щось змінити, то після натискання кнопки в панелі адміністратора «Опублікувати» зміни вступлять в силу. Модель автоматично робить оновлення та збереження даних.

У нас залишається ще дві моделі. У FG_PostTypeModel знаходиться перелік правил для групи полів, яку створює користувач. Оскільки це складний тип, який містить в собі різні поля, то необхідні враховувати всі нюанси. Модель містить ряд правил, наприклад, опис групи, відображення в панелі адміністратора, вивід іконки та інше. Все це робить групу більш комфортною для користувача в роботі.

У FiledPostTypeModel описані правила для конкретних полів. Коли користувач створює групу полів в ній міститься певна кількість звичайних полів різного типу. Модель містить правила для назви поля, його опису, видачі його при пошуку і іншого. Всі дані передаються в контролер FiledPostTypeController і вже там обробляються в момент наповнення полями конкретної групи полів.

Можливо, в подальшому для інших можливостей плагіна будуть розроблені нові моделі. На даний момент цих цілком достатньо, щоб розширення працювало з WordPress і здійснювало свої основні функції.

7.4 Уявлення

Незважаючи на те що плагін є розширення WordPress з готовою панеллю адміністратора, де є готовий призначений для користувача інтерфейс, необхідно вручну створювати уявлення для налаштувань. Патерн MVC допомагає зручно розділяти створення призначеного для користувача інтерфейсу і серверну частину. Саме тому було прийнято рішення створити уявлення призначеного для користувача інтерфейсу налаштувань, а не впроваджувати верстку всього цього в CMS WordPress.

Для того, щоб користувач міг вибрати формат виведення записи і вибрати рубрику, до якої буде прив'язана запис розроблено уявлення output_settings.php. Щоб вибірка була максимально зручною використовуються теги <select> мови HTML[9]. Все що потрібно

користувачеві – це зайти в панель адміністратора і вибрати потрібні йому варіанти. Далі зберегти зміни, після чого вони будуть опубліковані на сторінці в браузері.

Для відображення групи полів в плагіні використовується метод під назвою post_metaboxes.php. Тут описано в якому вигляді формується блок з групою полів, а також висновок кожного поля окремо. Щоб зробити повноцінну групу полів в неї потрібно додати поля конкретних типів. При створенні кожного поля воно отримує назву, вказану користувачем, вона відображається в полі label, а також створюється саме поле, в яке буде записано значення поля в input.

Плагін допомагає вибрати формат виведення записи і під кожен з варіантів написана своя верстка. Всі шаблони зберігаються в поданні template_page.php. Саме тут в базу даних потрапляє верстка конкретного шаблону, який раніше був обраний користувачем. Також тут описані всі існуючі формати виведення в плагіні. Надалі файл буде розширюватися новими варіантами, коли це буде розроблено. Швидше за все їх стане набагато більше, щоб вирішити якомога більше рутинних завдань по верстці.

Розглянувши всі уявлення, які пов'язані з функцією плагіна по вибору формату виведення переходимо до створення групи полів. Уявлення, які відповідають за цю частину знаходяться в папці field_groups.

Відразу варто виділити уявлення fg_metabox.php. У ньому відбувається відображення призначеного для користувача інтерфейсу для створення полів. Тобто, коли користувач почне наповнювати групу полів, він зможе вказати різні налаштування для конкретного поля. Все це описано в поданні і зроблено максимально зручно, щоб звичаний користувач розумів, де і що йому вписувати.

Також в цьому уявленні реалізований інтерфейс для динамічного оновлення інформації в полях і їх видалення з групи. Таким чином, в одному місці користувач зможе робити все що завгодно з полями різних типів.

Далі у нас уявлення locations_metabox.php. Тут описаний інтерфейс для того, щоб користувач міг додавати групу полів в конкретну рубрику записів на сайті. Надалі група

полів буде виведена на цільовій сторінці, тут же є підказки, які допоможуть користувачеві зрозуміти, що робити.

Подання відповідає ще й за те щоб до групи полів можна було додавати налаштування. Цей інтерфейс допоможе якось налаштувати групу полів і вже з обраним функціоналом, вона з'явиться в панелі адміністратора на конкретній сторінці.

На цьому ключові уявлення закінчуються. Для плагіна був розроблений максимально зручний призначений для користувача інтерфейс для проведення налаштувань. Таким чином, плагін ще й має інтуїтивно зрозумілий інтерфейс, що буде зручно для користувача, який раніше ніколи не мав сайту і не розуміє як з ним працювати через панель адміністратора.

8 РОБОТА ПЛАГІНУ

Для того, щоб протестувати роботу розширення необхідно встановити CMS WordPress і завантажити плагін. Після цього в розділі "Плагіни" користувачеві потрібно знайти «Post_format_view» і активувати його. Після цих дій плагін буде запущений на сайті.

8.1 Налаштування для роботи з контентом

Найкраще використовувати плагін «Post_format_view» для розробки нового сайту. В такому випадку розробник зможе спочатку розробити структуру для контенту і надасть власникові сайту масу можливостей для його редагування.

Насамперед після активації плагіна необхідно вирішити, який контент на сторінках буде статичним, а який буде представлений з використанням плагіна і може динамічно змінюватися. Як тільки буде вибрано кілька типів контенту необхідно зробити для них окремі рубрики з інтуїтивно зрозумілими назвами. Робиться це через розділ «Записи» - «Рубрики». Далі заповнюємо стандартні поля для нової рубрики і додаємо її на сайт.

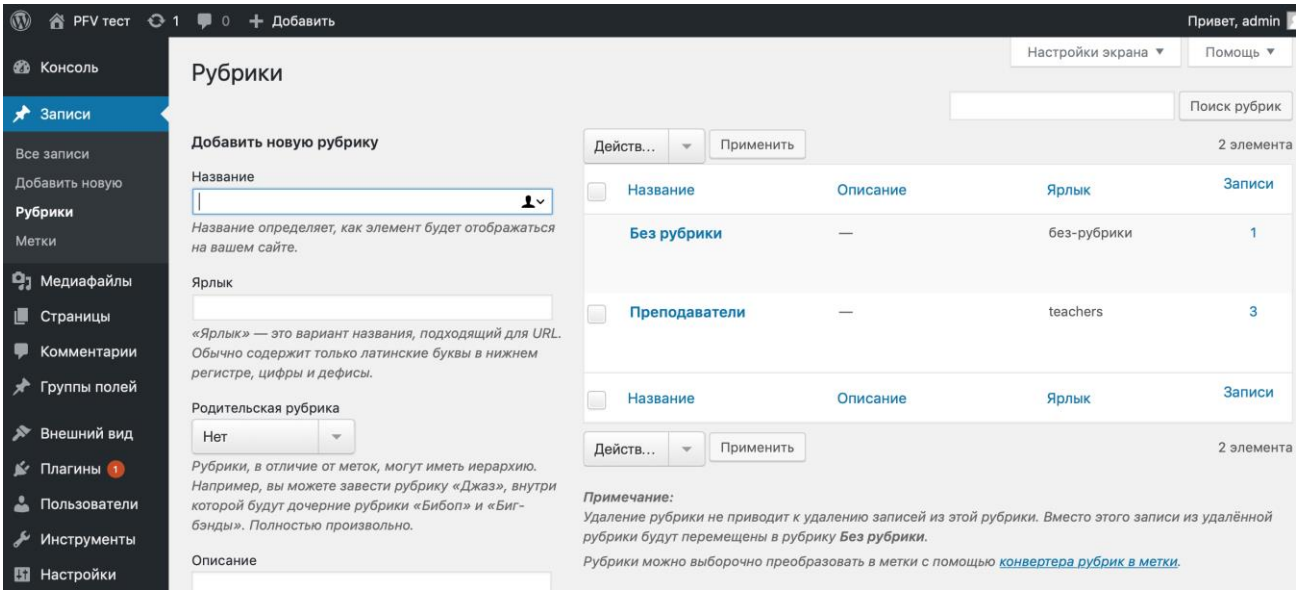


Рисунок 8.1 – Додавання нової рубрики

Рубрики потрібні для того, щоб розділити контент за змістом. Наприклад, на новинному порталі є новини, сторінки, де виведені автори новин, загальний контент за розділами і багато іншого. Плагін допоможе реалізувати записи з унікальною групою довільних полів під новини. Також можна зробити іншу групу полів під картки авторів новин, де контент буде вже зовсім іншим. Для загальної інформації по розділах тих же новин ми знову створюємо нову рубрику, окрему групу полів і тільки потім виводимо записи такого типу.

Подібне плагін дає можливість робити з чим завгодно. У підсумку виходить унікальний контент зі своїми конкретними полями, разом з цим вони редагуються з панелі адміністратора, можна вибрати на якій сторінці їх виводити і визначити для конкретного типу записів формат виведення – це вже налаштування візуальної складової виведення.

Після створення рубрик необхідно зробити потрібну кількість записів для користувача. Це робиться у розділі «Записи» - «Додати нову». Надалі їх можна додавати ще, але не варто забувати, що при створенні необхідно вказувати конкретну рубрику.

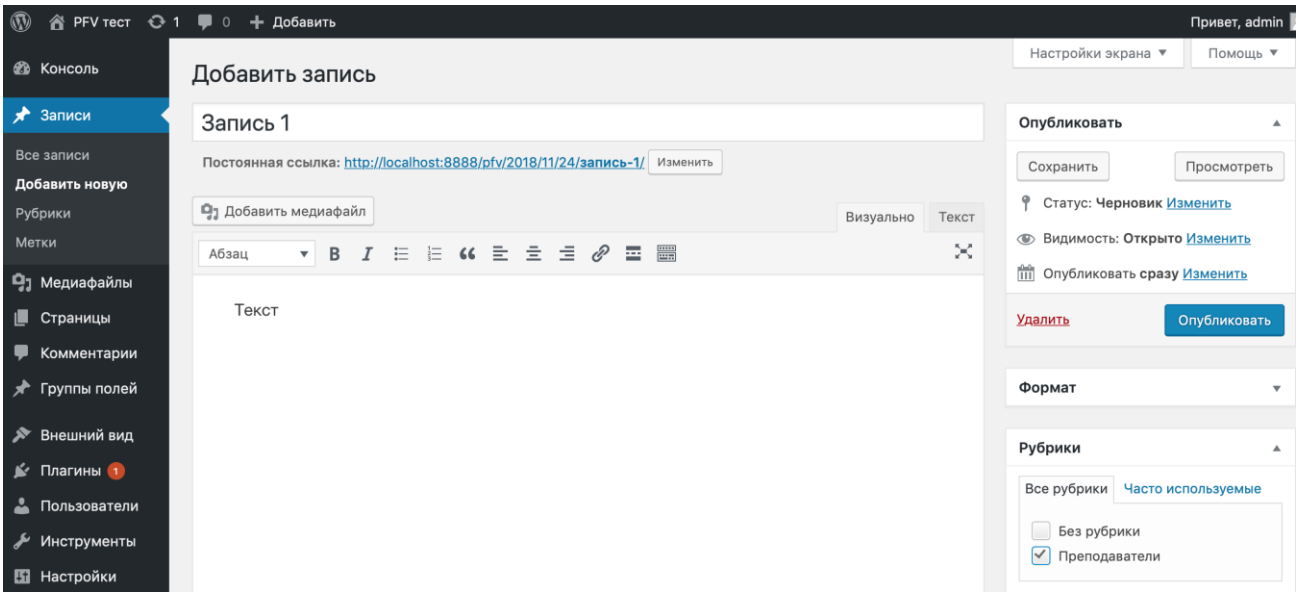


Рисунок 8.2 – Додавання нових записів

Теперт переходимо до роботи з плагіном «Post_format_view». Для початку в панелі адміністратора потрібно перейти в розділ «Групи полів» - «Додати групу полів». Тут же є і підрозділ «Групи полів», де можна подивитися всі існуючі групи.

Можна створювати поля для групи типу: текст, текстове поле, вибірка, check-box, варіанти відповіді, числове поле і картинка (на даний момент це всі типи, але їх кількість буде збільшуватися). За допомогою зручного інтерфейсу створюється необхідний набір полів для конкретного різновиду записів. Потім групі присвоюється ім'я для того, щоб розуміти за що вона відповідає і вибирається налаштування розташування виведення – це рубрика, до якої буде відноситися група полів. Залишається тільки опублікувати групу що створили. Виглядає в панелі адміністратора[10] це так:

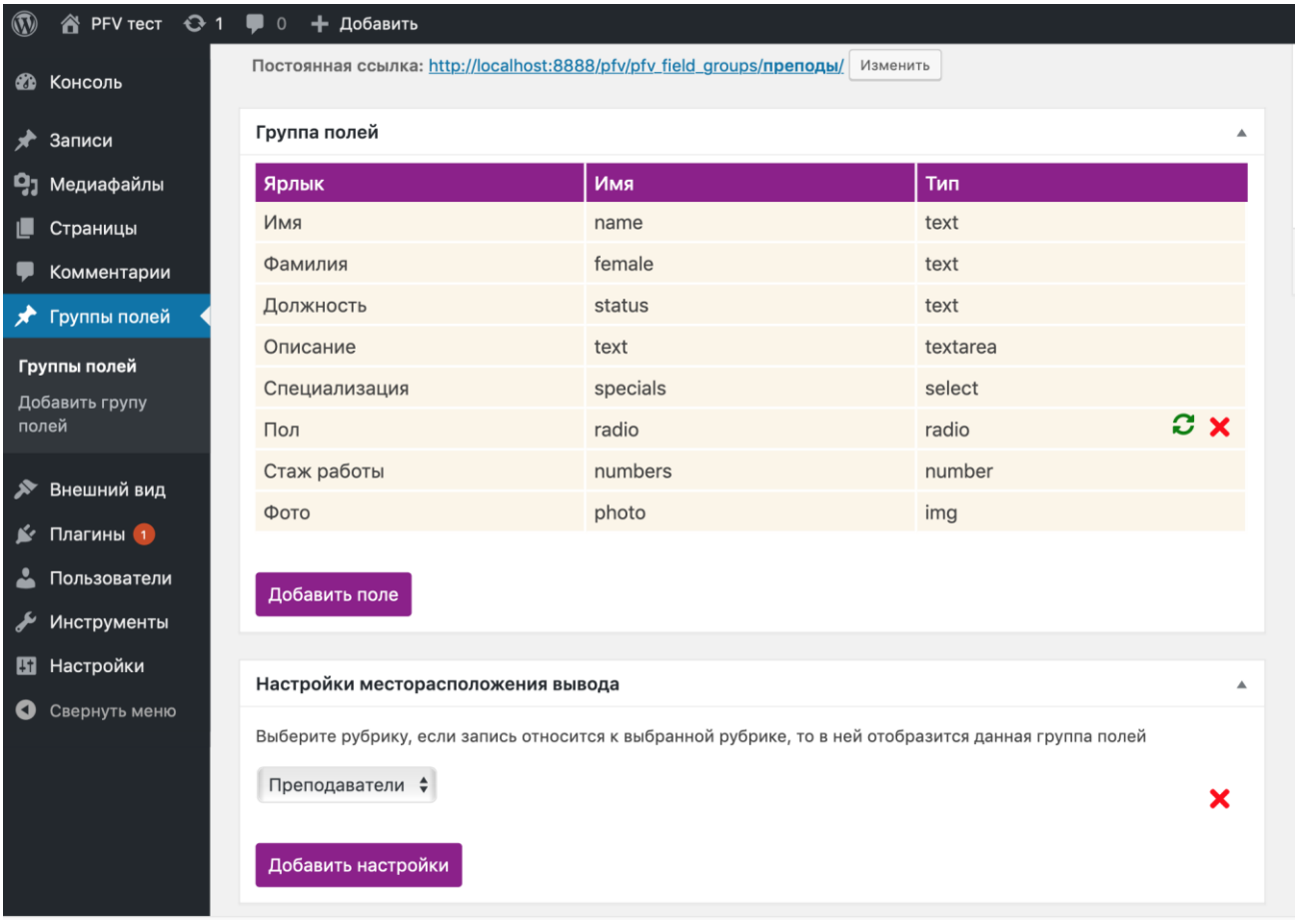


Рисунок 8.3 – Группа полів

Тепер можна перейти до запису з рубрики, до якої була додана група полів і побачити, що поля успішно додані. Їх можна заповнювати з панелі адміністратора і тим самим створювати потрібний контент. У записі група полів виглядає так:

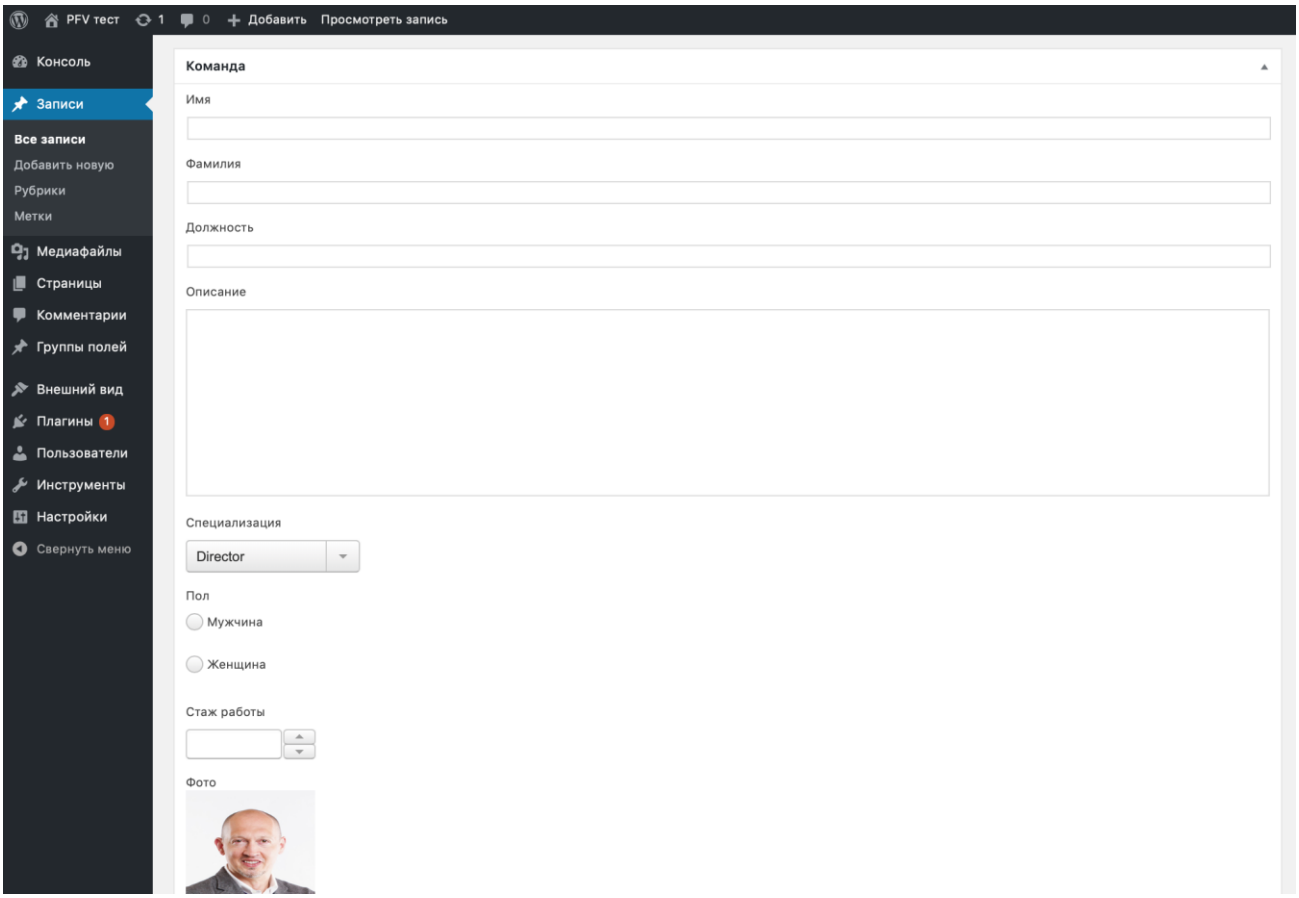


Рисунок 8.4 – Группа полів

Останнім кроком буде вивід контенту на конкретній сторінці. У WordPress сайти робляться за допомогою статичний сторінок, до яких застосовується шаблон верстки, який вибрав користувач в панелі адміністратора. Плагін робить прив'язку рубрики із записами до конкретної сторінки, де буде виводитися контент. При цьому в панелі адміністратора при редагуванні сторінки з цими записами можна вибрати формат виведення. Це заключний етап роботи з плагіном, який дає можливість відобразити записи в потрібному вигляді. Вибирайте 1, 2, 3, 4, 5 колонки по grid сітці[11; 12] або слайдер (надалі кількість форматів виводу буде збільшено).

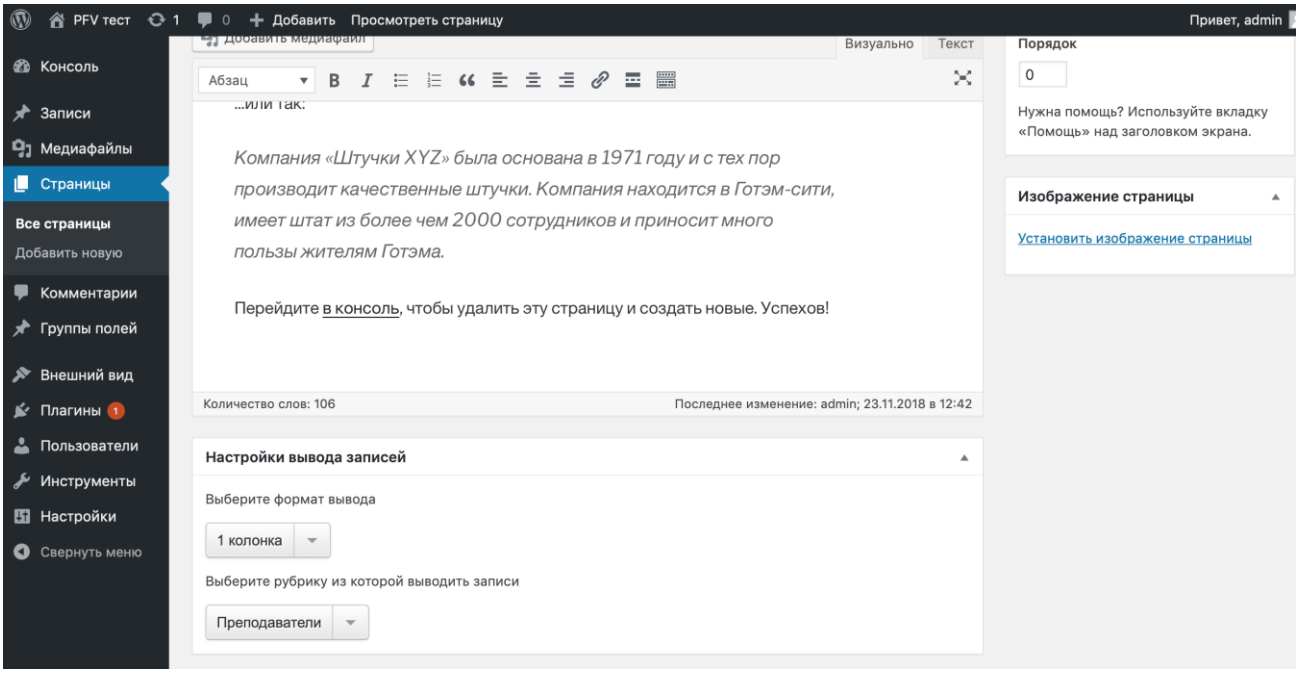


Рисунок 8.5 – Вибір формату виводу

Після цього можна переходити на сайт і дивитися, що вийшло. Найкраще щоб роботу з плагіном здійснював розробник сайту, який спочатку придумає правильну структуру для контенту. Це і буде автоматизацією проектування сайту на CMS WordPress. Після здачі сайту в експлуатацію власнику, він зможе самостійно додавати записи, редагувати їх завдяки даним в полях і вибирати формат виведення. Інтуїтивно зрозумілий інтерфейс дасть можливість це робити не звертаючись до розробника.

8.2 Результат роботи плагіну

Для демонстрації прикладу роботи плагіна створимо сторінку, де розташовані співробітники якоїсь компанії. Подібні сторінки присутні практично на всіх сайтах, тому цей приклад буде доречний. Спочатку створюється рубрика під назвою «Команда», а також під неї формується група полів з такою ж назвою, для того, щоб було зрозуміло до якого різновиду записів відноситься група. Після цього створюємо 5 співробітників і заповнюємо довільні поля.

Залишається тільки перейти на створену сторінку раніше і вивести на ній записи з членами команди. Також потрібно вибрати один із запропонованих форматів виводу. Результат виведення записів з членами команди в різних форматаха виведення виглядає так:

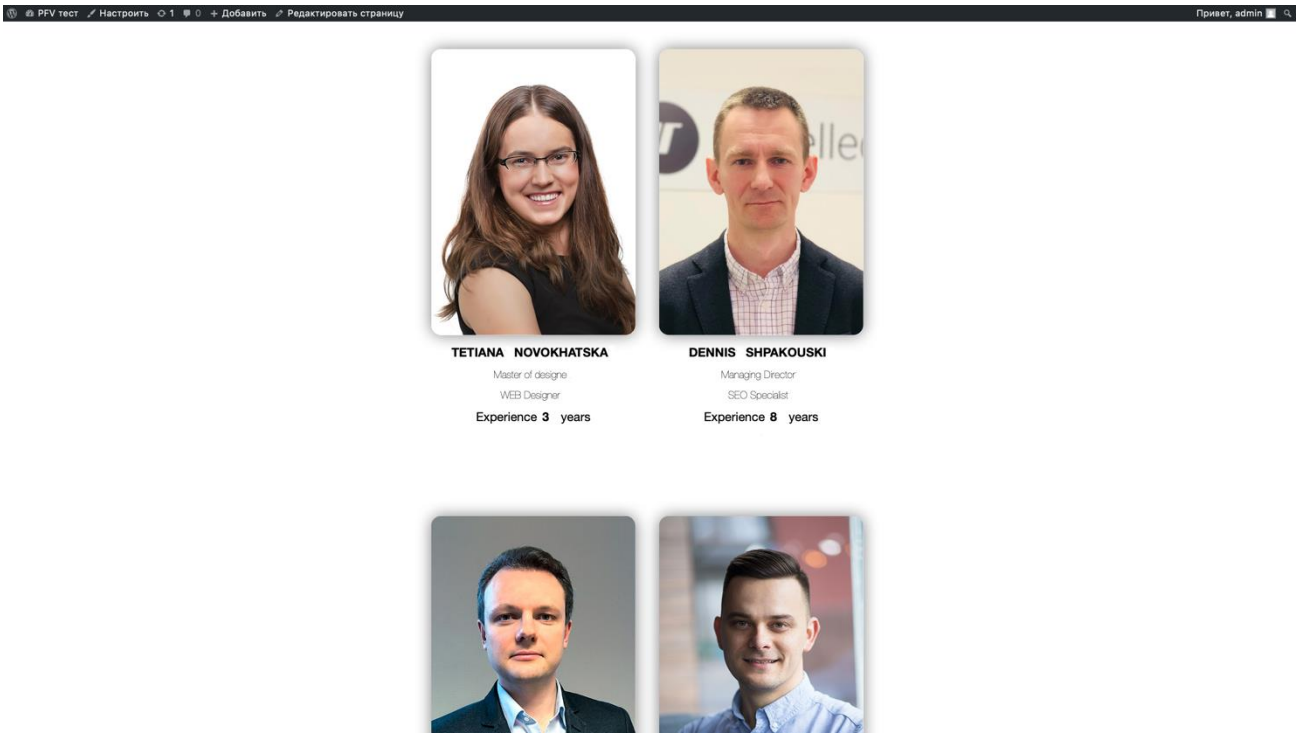


Рисунок 8.6 – Формат виводу 2 колонки

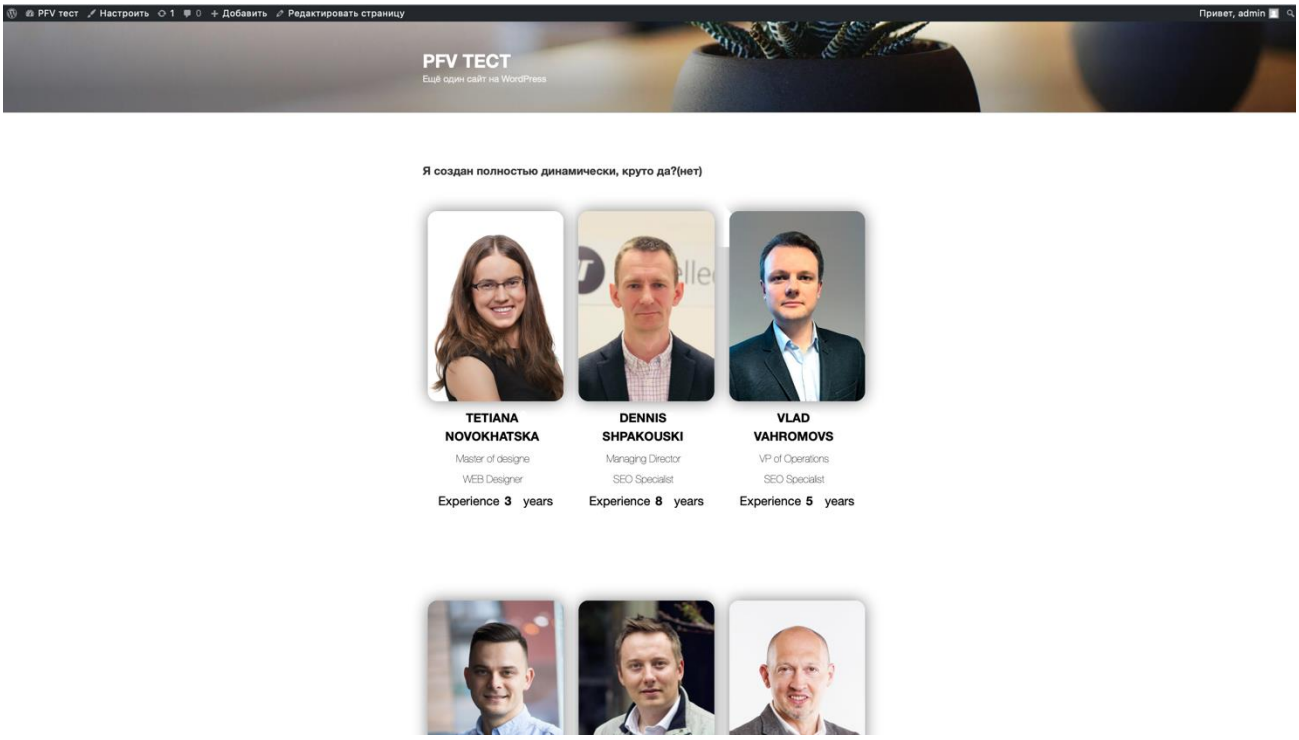


Рисунок 8.7 – Формат виводу 3 колонки

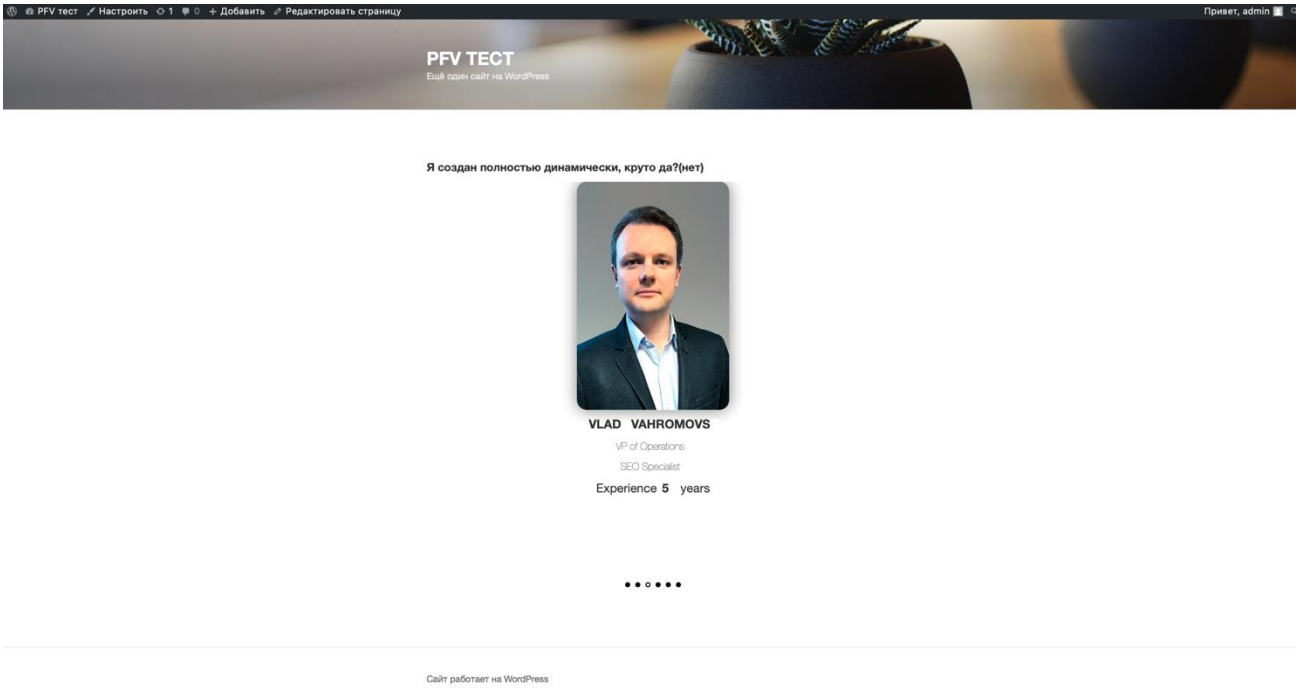


Рисунок 8.8 – Формат виводу слайдер

8.3 Майбутнє плагіну

Розробники WordPress ще з моменту запуску своєї системи керування вмістом створили сховище плагінів – WordPress Plugin Directory[13], куди кожен розробник міг викласти своє розширення. Щоб це зробити необхідно правильно оформити плагін, після чого завантажити його і надалі займатися розвитком власної розробки якщо це буде потрібно. На даний момент в офіційному репозиторії плагінів WordPress понад 55 тисяч розширень і ця цифра зростає щодня.

Всі плагіни від розробників потрапляють в сховище в безкоштовному вигляді. Проте кожен програміст може перетворити своє розширення для WordPress[14] в стартап додавши платну підписку на використання плагіна або платну версію, яка має розширені можливості.

Оскільки плагін, який розробляється для магістерської дисертації буде доповнюватися тривалий час, то з'явилася ідея в подальшому зробити ще й платну версію. Справа в тому, що поступово розширення почне обростати новими типами полів, а також новими форматами виведення даних. Все це робиться для того, щоб максимально позбавити розробника від рутинної роботи, а власнику сайту надати максимум можливостей з управління контентом.

Всі плагіни від розробників потрапляють в сховище в безкоштовному вигляді. Проте кожен програміст може перетворити своє розширення для WordPress в стартап додавши платну підписку на використання плагіна або платну версію, яка має розширені можливості.

На даний момент не було думок щодо того буде це платна версія або платна підписка на використання плагіна в цілому, але скоріше за все в підсумку буде обраний 1 варіант. В такому випадку безкоштовна версія плагіна буде пропонувати власнику сайту роботу з примітивними типами полів на кшталт рядки тексту, списків, картинок і надавати для застосування кілька простих форматів виводу. Для платної версії плагіна будуть розроблені більш складні типи полів, наприклад, це можуть бути різноманітні таблиці з можливість редагувати кількість рядів і стовпців для занесення інформації.

Разом з цим додатися кілька форматів виведення таблиць, як за структурою, так і за стилістикою, що теж буде дуже зручно.

Також можна додати поля типу вкладок. Це дасть можливість робити блоки, які будуть складатися з вкладок з заголовками і контентом в блоці під ними. При перемиканні відбуватиметься зміна блоку. Форматів виведення під такий тип полів також може додатися. Насправді варіантів доопрацювання плагіна маса, тому реалізація в майбутньому платної версії має сенс.

Опублікувати розроблений плагін для WordPress варто через наступні причини:

- розробник бачить необхідність в тих функціях, які ще раніше не були представлені в інших плагінах;
- використовуваних раніше плагінів недостатньо для комфортної розробки сайту і роботи з контентом на них в подальшому;
- спробувати внести свій вклад в розвиток WordPress;
- спробувати в майбутньому зробити платну версію для отримання прибутку.

Як правило, саме з цих причин розробники і створюють власні плагіни для системи керування вмістом WordPress.

Як правильно викласти свій плагін?

Першим кроком буде ознайомлення з докладним керівництвом[15] від розробників WordPress про те, що повинно бути в плагіні перед відправкою на розгляд. Основні положення:

- плагін потрапляє під ліцензію GPL;
- користувач погоджується з публікацією плагіна;
- розширення не містить спаму;
- немає дій, які є образливими або незаконними;
- відсутня вкладені посилання на сторонні сайти.

Далі потрібно перевірити в офіційному репозиторії плагінів чи немає збігів за назвою плагіна. Навіть якщо ви їх не знайдете є ймовірність того, що плагін з такою

назвою вже в розробці, але ще не викладений. У такому випадку потрібно просто зробити це першим.

Також необхідно заповнити файл readme, щоб в директорії плагіна WordPress з'явилися дані про нього. Наприклад, ви повинні вказати ім'я плагіна, авторів, прикріпити теги, вказати останню версію, ліцензію і зробити короткий опис. Для правильного офомлення є стаття "Creating Awesome WordPress.org Pages for your Plugin"[16], яка використовується багатьма розробниками.

Останнім етапом буде подати плагін на розгляд. Для початку створюється обліковий запис WordPress. Після цього заповнюється вся необхідна інформація для адміністрації і прикладаються файли з розроблених плагіном. Залишається тільки дочекатися відповідь, яка, як пишуть творці WordPress може зайняти максимум 8 годин. В кінці розробнику надсилають лист зі схваленням чи відмовою. Якщо другий варіант, то докладно розписують, як змінити ситуацію в позитивну сторону і заявку на схвалення можна відправити повторно.

Також варто згадати про інструмент Subversion, який дозволяє керувати версіями плагіна і відстежувати зміни в ньому. При необхідності з його допомогою можна переходити назад на старі версії. Система схожа на Git.

Коли плагін схвалюють творці WordPress Subversion repository можна зберегти його копію. Subversion надалі буде використовуватися, як сховище файлів. Копію файлів зможе отримати будь-який користувач, але вносити зміни в актуальну версію тільки розробник плагіна.

Як тільки плагін буде остаточно закінчено і постане питання про те, як правильно викладати його в офіційне сховище плагінів WordPress, то інструмент Subversion буде вивчений більш детально. На даний момент в цьому немає необхідності.

Також готова безкоштовна версія плагіна скоріше за все буде оформлена за стандартами, які описані творцями WordPress і розміщена в репозиторії, щоб подивитися на відгуки інших користувачів. Це дасть можливість заглибитися в функціонал плагіна і зрозуміти які у нього є проблеми, а також, можливо, запровадити якісь нові функції, які

будуть корисні для користувачів. Спільнота WordPress надасть масу варіантів поліпшення плагіна і варіантів його подальшого розвитку.

9 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

9.1 Опис ідеї проекту

9.1.1 Зміст ідеї, що пропронується, можливі напрямки застосування та основні вигоди, які зможе отримати користувач товару, описані у таблиці 9.1.

Таблиця 9.1 – Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
	1. Розробка сайтів на WordPress	Підвищення швидкості розробки сайту на WordPress та автоматизація його використання в майбутньому
	2. Управління готовимми сайтами на WordPress	Розришення можливостей для роботи з контенто на готовому сайті на WordPress

9.1.2 Аналіз потенційних техніко-економічних переваг ідеї

Таблиця 9.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Плагін ACF*	*Плагін Elementor*	*Плагін Page Builder*			
1.	Технологічна собівартість товару (грн)	560	700	1372	812	-	-	+
2.	Кількість разів використання (шт. сайтів)	Безліміт	1	1	1	-	-	+

9.2 Технологічний аудит проекту

Таблиця 9.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
	Програмні засоби для автоматизації розробки сайтів на WordPress та розришерння можливостей для работи з конентом	Програмний меод реалізації	Технології наявні	Технології доступні
Обрана технологія реалізації ідеї проекту: Програмний метод реалізації проекту.				

9.3 Аналіз ринкових можливостей запуску стартап-проекту

9.3.1 Аналіз попиту

Таблиця 9.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	стагнує
2	Загальний обсяг продаж, грн/ум.од	стагнує
3	Динаміка ринку (якісна оцінка)	зростає
4	Наявність обмежень для входу (вказати характер обмежень)	зростає
5	Специфічні вимоги до стандартизації та сертифікації	спадає
6	Середня норма рентабельності в галузі (або по ринку), %	80%

Таким чином, можемо зробити висновок, що ринок є привабливим для проекту.

9.3.2 Потенційні групи клієнтів

Таблиця 9.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
	Можливість автоматизувати редагування та виведення контенту на готових веб-сайтах.	Всі бізнесмени, які продають свої товари або послуги через веб-сайт на WordPress	Поведінку клі*єнта формують особливості, складність та обсяг інформаці*ї на сайті	Зменшення собівартості програмного рішення

9.3.3 Аналіз ринкового середовища

Таблиця 9.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Ускладнення роботи з WordPress	Проблеми в роботі з контентом	Підвищення собівартості программного рішення

Таблиця 9.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Збільшення кількості подібних розробок	Поява нових плагінів для роботи з контентом	Оновлення плагіну для боротьби з новими конкурентами

9.3.4 Аналіз пропозиції

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Тип конкуренції Чиста конкуренція	Відсутність монополій та конгломератів	Підвищення якості послуги. Підвищення якості обслуговування.
2. За рівнем конкурентної боротьби: Локальний	Надання программного забезпечення на рівні одні системєї системи	Влосконалення роботи программного забезпечення
3. За галузевою ознакою: Внутрішньогалузева	Конкуренте середовище полягає у забезпеченні власників сайтів розрищеними можилвостями	Розришерння галузевого впливу
4. Конкуренція за видами товарів: Товарно-родова	Різноманітні шляхи задоволення конкретного бажання	Покращення якості та зменшення ціни

5. За характером конкурентних переваг Цінова	Зменшення собівартості продукції	Додавання нових можливостей без зміни ціни
6. За інтенсивністю: Не марочна	Немає своєї торгової марки	Розвиток своєї торгової марки

9.3.5 Аналіз умов конкуренції в галузі

Таблиця 9.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	*Плагіни які працюють з довільними полями. Плагіни конструктори сторінок*	*Патенти на программний продукт. Наявність товарних знаків*	*Значення розміру*	*Рівень чутливості до зміни цін*	*Бар'єри проникнення*
Висновки:	Середня інтенсивність конкурентної боротьби	є можливості входу в ринок є потенційні конкуренти	Постачальники не диктують	Клієнти диктують умови на ринку шляхом ускладнення сайтів	Можуть давади обмежену кількість можливостей

9.3.6 Перелік факторів конкурентоспроможності

Таблиця 9.10 – Обгрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
	Економічний фактор	Вартість программного забезпесення
	Рівень технології виробництва	Досконаліть виробничого виконная
	Попит	Сайти стають все більш складнішими і власникам необхідно комфортно з ними правцювати після розробки

9.3.7 Аналіз сильних та слабких сторін стартап-проекту

Таблиця 9.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з ... (назва підприємства)*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Економічний фактор	18		+					

2	Рівень технології виробницта	15				+			
3	Попит	14		+					

9.3.8 Матриця аналізу сильних (Strenght) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities)

Таблиця 9.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Низька собівартість проекту Багаторазове використання Простота використання	Слабкі сторони: Відсутність торгових марок та патентів Невелика кількість можливостей на данному етапі розробки
Можливості: Збільшення кількості можливостей Розширення галузевого впливу	Загрози: Зниження потреби в сайтах на WordPress Товарно-родова конкуренція

9.3.9 Альтернативи ринкової поведінки

Таблиця 9.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс*	*Ймовірність отримання ресурсів*	*Строки реалізації*

	заходів) ринкової поведінки		
1	Низька вартість товару	0,7	1-2 роки
2	Вирішення більшої кількості проблем користувача	0,5	2-3 роки

З зазначених альтернатив обираємо другу: з більшою ймовірністю зменшення сроків реалізації та вдосконалення на основі готових ресурсів.

9.4 Розроблення ринкової стратегії проекту

9.4.1 Визначення стратегії охоплення ринку

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Розробка сайтів на WordPress	Висока	Середній	Низька	Середня
Які цільові групи обрано: розробка сайтів на WordPress					

9.4.2 Базова стратегія розвитку

Таблиця 9.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1	Цільовою групою є сайти розроблені на WordPress	Стратегія концентрованого маркетингу	Низька собівартість програмного забезпечення	Стратегія лідерства по витратах

9.4.3 Вибір стратегії конкурентної поведінки

Таблиця 9.16 – Визначення базової стратегії конкуретної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
	Ні	Буде шукати нових споживачів та забирати існуючих	Не буде	Стратегія заняття конкуретної ніші

9.4.4 Стратегія позиціонування

Таблиця 9.17 – Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1	Зменшення собівартості програмного забезпечення	Стратегія лідерства по витратах	Технічна собівартість Багаторазове використання	Низька ціна Вирішення проблем Постійне використання

9.5 Розроблення маркетингової програми стартап-проекту

9.5.1 Формування маркетингової концепції проекту

Таблиця 9.18 – Визначення ключових переваг концепції потенційного проекту

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
	Більш гнучке управління контентом на сайті	Постійне вирішення проблеми Низька собівартість	Низька ціна Багаторазове використання

		Високий рівень технічної реалізації	

9.5.2 Трирівнева маркетингова модель товару

Таблиця 9.19 – Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Розщирення можливостей для редагування контенту на готовому веб-сайті		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	Технологічна собівартість товару	М	Тл
	Якість: стандарт WordPress, ліцензія GPL		
	Пакування: відсутнє		
	Марка: кафедра АУТС КПІ ім. Сікорського		

III. Товар із підкріпленням	До продажу: Гарантія якості
	Після продажу: Оновлення товару безкоштовно
За рахунок чого потенційний товар буде захищено від копіювання: Захист інтелектуальної власності	

9.5.3 Визначення цінових меж

Таблиця 9.20 – Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
	-	700-1400	20000-30000	500-600

9.5.4 Визначення оптимальної системи збуту

Таблиця 9.21 – Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1	Купівля розширення перед початком розробки веб-сайту	Вставнолення безпосередніх контактів із	Канал нульового рівня, що складається з	Власна система збуту, що забезпечить обслуговування

		споживачами та покупцями товарів Участь у формуванні ціни Обслуговування клієнтів	виробника, який продає свій товар беспосередньо споживачам	товару, формування ціни та встановлення контактів зі споживачами

9.5.5 Розроблення концепції маркетингових комунікацій

Таблиця 9.22 – Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуютьс я цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*

	Купівля розробки веб-сайту для тривалого використання	Персональні Інформаційні	Позиціонування по перевазі	Інформуюче Переконуюче	Представлення вигоди товару

9.6 Висновки

Ідеєю даного стартап-проекту є створення програмних засобів для автоматизації розробки сайтів та розришерння можливостей для работи з конентом. Напрямком застосування є система керування вмістом WordPress.

Є можливість ринкової комерціалізації проекту, оскільки наявний попит та спостерігається динаміка ринку.

Конкуренція на ринку чиста, внутрішньогалузева та товарно-родова, що сприяє входженню товару на ринок.

Як альтернативу впровадження доцільно обрати низьку ціну товару та розришення можливостей товару.

Таким чином, можемо зробити висноков, що подальша імплементація проекту є доцільною.

ВИСНОВОК

У магістерській дисертації був розроблений плагін для автоматизації розробки веб-сайтів на WordPress і управління контентом. Основне завдання плагіна надати власнику сайту широкий спектр можливостей для редагування контенту і вибір формату виведення.

В процесі розробки плагіна для WordPress була вибрана мова програмування php, оскільки на ньому створюється більшість розширень для цієї системи керування вмістом.

Для створення інтерфейсу користувача в панелі адміністратора, який дозволяє налаштовувати плагін використовувалися мови html та css[17]. Деякі елменти призначеного для користувача інтерфейсу створювалися з застосування JavaScript[18; 19] і технології AJAX.

Розробка плагіна велася з урахуванням всіх стандартів WordPress, які описані на офіційному сайті. Оформлення розширення також відповідає цим стандартам.

Для того, щоб спростити розробку плагіна і отримати можливість максимально просто його розширювати в майбутньому використовувався патерн проектування MVC[20; 21]. Концепція, в якій присутні моделі, контролери та уявлення ідеально підійшла для даної розробки. У моделях описані необхідні для роботи плагіна дані і методи роботи з ними. У контролерах описана логіка для дій плагіна після впливу на нього з боку користувача. В уявленнях реалізовані шаблони виведення даних в браузер для користувача.

Структуру плагіна умовно можна розділити на 2 частини. Перша відповідає за створення групи полів і роботу з нею. Друга описує логіку для вибору формату виведення контенту. Все це розділене зручним чином, оскільки плагін має дві основні функції.

За час роботи над магістерською дисертацією в плагіні вдалося реалізувати створення групи полів і роботу з нею, а також динамічний вивід контенту за допомогою форматів виводу.

Для групи полів було разроблено кілька типів полів, щоб можна було через панель адміністратора відразу вибирати певні варіанти контенту. На даному етапі є можливість додавати рядок тексту, абзац, картинку, список і варіанти відповідей. Плагін розпізнає всі ці типи і виводить їх на сторінці в потрібному вигляді.

Також реалізовано створення різних типів записів, які відрізняються в залежності від обраної рубрики. На статичній сторінці можна вибирати створений користувачем шаблон і виводити групу полів в необхідному місці з потрібними налаштуваннями та інформацією в довільних полях.

Формати виведення контенту також реалізовані. На даний момент це поділ тексту і інших типів полів на певну кількість колонок. По суті імітація сітки grid, що позбавляє власника сайту від потреби робити нову верстку. Також за допомогою плагіна можна зробити слайдер із записів, які додав користувач раніше.

Надалі плагін буде розширюватися і в ньому з'явитися більше типів полів для того, щоб можна було додавати контент різного характеру. Також будуть додані формати виведення для того, щоб позбавити програмістів від рутинної роботи. Всі описані завдання в магістрескій дисертації були вирішені, але робота над плагіном триває.

ПЕРЕЛІК ПОСИЛАНЬ

1. Переваги WordPress - https://habr.com/post/251315/

2. Кодекс WordPress - https://codex.wordpress.org/Database_Description

3. WordPress Plugin Directory - https://wordpress.org/plugins/

4. Руковдство разработчику плагина - https://wordpress.org/plugins/developers/

5. Статья Creating Awesome WordPress.org Pages for your Plugin - https://www.sitepoint.com/create-awesome-wordpress-org-page-plugin/

6. WordPress Plugin Boilerplate - https://wppb.me

7. Основы шаблонов.- https://codex.wordpress.org/Основы_шаблонов

8. Введення в системи баз даних, 2003 г. - К. Дж. Дейт

9. HTML і XHTML. Детальне керівництво, 2002 г. - Муссіано, Кеннеді

10. CSS рецепти програмування, 2015 г. - Крістофер Шмітт

11. Web-дизайн по стандартам, 2007 року - Зельдман Д.

12. Досконалий код. Практичний посібник з розробки програмного забезпечення, 2005 р - Макконнелл С.

13. JavaScript. Детальний керівництво, 2008 - Фленаган Д.

14. jQuery. Докладне керівництво по просунутому JavaScript, 2008 - Бібо Б., Кац І

15. Прийоми об'єктно-орієнтованого проектування. Патерни проектування, 2014 г. - Гамма, Хелм, Джонсон, Вліссідес

16. Чуйний веб-дизайн, 2011 р - Ітан Маркотт

17. Мова UML. Керівництво користувача (2-е изд.), 2006 р - Буч Г., Рамбо Д., Якобсон І.

18. Робота з Bootstrap – Режим до ресурсу: http://mybootstrap.ru/get-started/

19. Адаптивна розробка сайтів – Режим до ресурсу: https://tproger.ru/translations/responsive-web-design-tips/

20. WordPress для професіоналів. Розробка та дизайн сайтів – Б. Уильямс, Д. Дэмстра, Х. Стэрн.

21. Хуки в WordPress - https://wp-kama.ru/hooks.

22. Довільні поля WordPress - https://codex.wordpress.org/Произвольные_поля.

ДОДАТОК А – Стаття

ЗМІСТ/CONTENTS

Використання патерна MVC для автоматизації проектування інтерактивної прикладної системи у Web

Тучин Владислав
КПІ ім. Ігоря Сікорського
Київ, Україна
tu4invladislav2511@gmail.com

Дорошенко Анатолій
КПІ ім. Ігоря Сікорського
Київ, Україна
doroshenkoanatoliy2@gmail.com

Анотація. **Метою даної роботи є використання популярного паттерна MVC для створення інтерактивної системи пошуку закладів громадського харчування. Для реалізації була використана платформа ASP.NET Core, щоб спростити вирішення поставлених задач. Шаблон MVC надає простий спосіб побудови структури додатка, метою якого є відділення бізнес-логіки від призначеного для користувача інтерфейсу. В результаті, додаток легше масштабується, тестується, супроводжується і звичайно ж реалізується. Завдяки існуючим можливостям і їх розширення вдалося розробити нову інтерактивну систему.**

Ключові слова: патерн проектування, MVC, модель, контролер, веб-сайт, інтерактивна система, пошук.

Вступ

Ідея створення інтерактивної системи пошуку закладів громадського харчування була викликана тим, що в інтернеті, як виявилося, досить мало сайтів з подібними можливостями. Жоден ресурс не надає всіх тих функцій, які закладені в представленому проекті. Першим гідним аналогом є пошук закладів за допомогою системи Google [1].

Сервіс пропонує зручну карту проїзду з можливістю прокласти маршрут, контактні дані закладу, кілька фотографій, графік відвідуваності, можливість залишати коментарі та список схожих закладів. З точки зору практичності для користувача інтерфейс сервісу від Google досить зручний, проте він не дає повноцінної інформації про ресторан. Немає загального опису того на чому спеціалізується місце, які послуги надає і яка в нього кухня. Сервіс обмежується простою градацією на кшталт «кафе», «караоке-бар» і «бар», при цьому не даючи можливості користувачеві зрозуміти, що в закладі ще є літня тераса, продаються кальяни в спеціально відведеній зоні і є інші послуги.

За детальними даними необхідно звертатися на офіційний сайт закладу. Також відсутній меню і ціни на страви. Не можна вивчити асортимент і зрозуміти чи доступний заклад з огляду на власне фінансове становище. Через сервіс можна тільки знайти на карті заклад, але вивчити його докладно не можна.

Створення сайту дозволяє залучати користувачів певного продукту або послуги, оскільки повністю відображає тематику і напрямок підприємства, що дозволяє залучати, а також взаємодіяти безпосередньо з цільовим споживачем. Так само розробка сайту - це спосіб розширення меж діяльності підприємства і освоєння нової маркетингової сфери – сфери Інтернету, найбільш унікальною для розвитку бізнесу. Сайт – це комерційний інструмент.

Існують готові інтерактивні рішення, які залучають власників закладів своєю простотою. Проте більшість з них не надають інструменти для гнучкого взаємодії зі сторінками та редагування. Саме тому описана інтерактивна система візьме на себе роль універсального інструменту з пошуку закладів громадського харчування, використовуючи в процесі різні критерії.

Для того, щоб створити таке програмне рішення використовувався патерн проектування MVC, який дав можливість максимально спростити процес розробки і зробити його більш ефективним для такої великої системи. Використовувані інструменти наочно показують наскільки зручніше виявляється MVC для створення сайтів-каталогів з різними функціями, ніж готове рішення на основі беслпатних CMS або самописних серверних частин за допомогою різних мов програмування.

Про патерн проектування MVC

Для реалізації була вибрана платформа для розробки ASP.NET Core [2], яка представляє собою технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів. З одного боку, ASP.NET Core є продовженням розвитку платформи ASP.NET. Але з іншого боку, це не просто черговий реліз. Вихід ASP.NET Core фактично означає революцію всієї платформи, її якісна зміна. ASP.NET Core тепер повністю є opensource-фреймворком. Всі вихідні файли фреймворку доступні на GitHub.

ASP.NET Core включає в себе фреймворк MVC [3], який об'єднує функціональність MVC, C# [4], Web API і Web Pages та використаний в реалізації інтерактивної системи. MVC – це не шаблон проекту, це конструкційний шаблон, який описує спосіб побудови структури нашого застосування, сфери відповідальності та взаємодія кожної з частин в даній структурі.

Вперше вона була описана в 1979 році, звичайно ж, для іншого оточення. Тоді не існувало концепції веб додатки. Tim Berners Lee (Тім Бернерс Лі) посіяв насіння World Wide Web (WWW) на початку дев'яностих і назавжди змінив світ. Шаблон, який ми використовуємо сьогодні, є адаптацією оригінального

шаблону до веб розробці. Ідея, яка лежить в основі конструкційного шаблону MVC [5] [6], дуже проста: потрібно чітко розділяти відповідальність за різне функціонування в наших програмах:

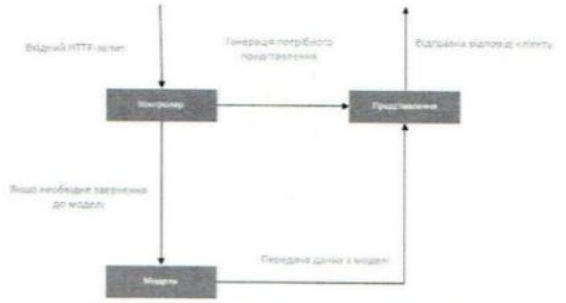


Рис. 1. Загальна схема взаємодії компонентів MVC

Додаток розділяється на три основних компоненти, кожен з яких відповідає за різні завдання. Контролер керує запитами користувача (одержувані у вигляді запитів HTTP GET або POST, коли користувач натискає на елементи інтерфейсу для виконання різних дій). Його основна функція – викликати і координувати дію необхідних ресурсів і об'єктів, потрібних для виконання дій, що задаються користувачем. Зазвичай контролер викликає відповідну модель для задачі і вибирає відповідний вид.

Модель – це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління додатком. У будь-якому додатку вся структура моделюється як дані, які обробляються певним чином. Тільки дані, які повинні бути оброблені відповідно до правил (дата не може вказувати в майбутнє, e-mail повинен бути в певному форматі, ім'я не може бути довшим X символів, і так далі). Модель дає контролеру уявлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти їх користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних. Модель містить найбільш важливу частину логіки нашого застосування, логіки, яка вирішує завдання, з якою ми маємо справу (форум, магазин, банк, тощо). Контролер містить в основному організаційну логіку для самого додатка (дуже схоже на ведення домашнього господарства).

Вид забезпечує різні способи представлення даних, які отримані з моделі. Він може бути шаблоном, який заповнюється даними. Може бути кілька різних видів, і контролер вибирає, який підходить якнайкраще для поточної ситуації. Веб додаток зазвичай складається з набору контролерів, моделей і видів. Контролер може бути влаштований як основний, який отримує всі запити і викликає інші контролери для виконання дій в залежності від ситуації.

Очевидною перевагою в даній інтерактивній системі, яку ми отримуємо від використання концепції MVC – це чіткий поділ логіки подання (інтерфейсу користувача) і логіки програми. Підтримка різних типів користувачів, які використовують різні типи пристроїв є спільною проблемою наших днів. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних. У даній ситуації це допомогло зробити систему адаптивної під різні пристрої і в залежності від того з чого користувач заходить на сайт контролер поверне йому відповідний для розв'язання екрану варіант дизайну. Крім ізолювання видів від логіки додатки, концепція MVC істотно зменшує складність великих додатків. Код виходить набагато більш структурованим, і, тим самим, полегшується підтримка, тестування і повторне використання рішень.

Поділ відповідальності і гнучкість дозволили налаштовувати різні компоненти платформи на свій розсуд. Змінювати будь-якої частини конвеєра роботи MVC або адаптувати його до своїх потреб. Відповідно, на відміну від таких безкоштовних CMS як Wordpress або Joomla, патерн проектування дав можливість реалізувати інтерактивну систему з усіма задуманими спочатку можливостями. У даній ситуації інформація про такі сутності, як міста, заклади, кухні, вулиці та інше стала розміщуватися всередині моделей. Взаємодія між ними разом з описом логіки вдалося розмістити в контролерах. За відображення інформації на сторінках відповідають уявлення.

Інструменти для створення систми пошуку

Вивчивши концепцію MVC вдалося розробити кілька інструментів, які допомогли значно простіше впоратися з реалізацією такого великого сайту-каталогу.

Моделі уявлення були розроблені як додатковий вид моделей для інтерактивної системи. Для найбільш зручної передачі даних в уявлення використовується ViewModel, яка називається моделлю уявлення. Вона використовується конкретно для виведення даних в одному уявленні. Також така модель задіюється якщо є необхідність редагувати певні сутності. У ситуації з відсутністю ViewModel під час редагування інформації на сторінці не відбудеться змін в базі даних. Коли таке відбувається то дані просто відображаються на екрані одного користувача, але при цьому редагування неможливе централізовано для всіх, результат не записується в базу даних.

Наявність ViewModel вирішило проблему редагування в данному проекті. Якщо нам необхідно було змінити певні властивості на сторінці, а не все відразу, то зручніше використовувати ViewModel для цих цілей. За допомогою такого класу модель уявлення містить тільки необхідні для редагування властивості. Для кожної сторінки інтерактивної системи були створені такі ViewModel. Вони допомогли значно простіше створити як інтерфейс для користувача, так і адміністратора. На прикладі моделі представлення окремого блогу можна подивитися, що за допомогою HomeBlogViewModel передаються обєкти відразу двох моделей для відображення:

```
public class HomeBlogViewModel
{
        public Blog Blog { get; set; }
 public virtual IEnumerable<BlogComment> Comments { get; set; }
}
```

Завдяки зручній моделі уявлення вдається передавати на сторінку блога відразу всі властивості класу Blog і BlogComment. Таким чином, подання буде показувати сам блог користувача і всі відгуки до нього від користувачів інтерактивної системи. Такий підхід дав можливість значно простіше створювати уявлення і виводити на них дані. При цьому в процесі редагування всі зміни потрапляють в базу даних і зберігаються.

Ще одним нововведенням стали елементи View Component, які називаються компонентами уявлень. По своєму функціналу вони дуже сильно схожі на часткові уявлення. При цьому ViewComponent використовувалися для таких завдань, які проблематично вирішувати за допомогою тільки часткових уявлень. До таких можна віднеси створення динамічного меню зі стравами та цінами, хмари тегів, панелі авторизації і т.п. У даній інтерактивній системі ViewComponent активно задіяні для контейнерів з різними додатковими даними, пошукових контейнерів, коментарів до закладів, меню та багатьох інших задач. Приклад застосування компонента уявлення для коментарів до закладу:

```
public class PlaceComments: ViewComponent
{
        private readonly Context _context;

        public PlaceComments(Context context)
    {
          _context = context;
        }
        public IViewComponentResult Invoke(PlaceComment item)
        {
         return View(new HomePlaceCommentsViewModel
{Comment = item });
        }
}
```

Компоненти уявлень складаються з двох частин: клас на C# і часткове уявлення, що викликає методи даного класу. ViewComponent не може генерувати HTTP-запити, а контент, який з їх допомогою генерується з'являється в коді батьківського уявлення. В інтерактивній системі використовується спосіб успадкування від базового класу ViewComponent для визначення компонентів. Щоб в правильній формі для даного проекту зробити подання вихідного результату використовується об'єкт інтерфейсу IViewComponentResult. Також для генерації уявлення Razor з можливістю передачі моделі існує вбудованих клас

ViewComponentResult, який використовується в компонентах уявлення інтерактивної системи. Щоб створити об'єкт даного класу використовується метод View(). У ситуації з успадкуванням класу компонента від базового класу ViewComponent зручно використовувати успадкований метод View. Як правило в проекті використовується версія перевантаження View(model), яка вибирає уявлення за замовчуванням і передає в нього необхідну модель для відображення контенту на сторінці, тому що це найбільш просте рішення при розробці.

При розробці використовувалася Visual Studio 2017 проекти якої мають вбудовану підтримку інструментів – Bower, Grunt, Gulp, які дали змогу управляти скриптами JavaScript [7] і стилями CSS [8], автоматизувати і оптимізувати процес веб-розробки системи. Для зберігання даних додаток використовує Microsoft SQL Server Express. Дане рішення, як правило використовується в зв'язці з ASP.NET MVC більшістю розробників за рахунок зручності і досить великого функціоналу. Сучасний Entity Framework дає можливість автоматично пов'язувати звичайні класи на C# з таблицями в базі даних. В інтерактивній системі здійснюється робота з MS SQL Server [9].

Для створення проекту використовувалося кроссплатформенне рішення Entity Framework Core 1.0 на базі .NET Core. Після створення нового проекту за шаблоном Web Application необхідно було підключити пакет Microsoft.EntityFramework Core.SqlServer. Далі підключити Microsoft.Entity FrameworkCore.Tools, який дасть можливість створити саму базу даних, щоб з нею можна було працювати надалі.

Для інтерактивної системи пошуку закладів громадського харчування використовується вбудована система аутентифікації і авторизації ASP.NET Identity. Завдяки даній системі користувач може створити обліковий запис на сайті, пройти аутентифікацію, управляти обліковими записами якщо мова йде про власника проекту та використовувати аккаунти зовнішніх провайдерів, наприклад, Google, Facebook, Microsoft та інших.

Адміністративна частина сайту представлена у вигляді самописної системи управління вмістом. На відміну від готових комп'ютерних програм (CMS) для створення сайту в проекті був створений спеціальний розділ Admin зі сторінками для конкретних налаштувань. Таким чином, адміністратор, увійшовши на сайт під своїм обліковим записом може вільно переміщатися по різних сторінками адміністративної панелі і здійснювати різного роду налаштування.

Інтерфейс користувача

Візуальна частина інтерактивної системи розроблялася з використанням розмітки сторінок HTML і стилів CSS, з використанням самописних скриптів на JavaScript з використанням бібліотеки JQuery [10] для вирішення певних завдань. У ASP.NET MVC для уявлення можна використовувати не тільки стандартний код HTML + CSS, а також вставляти код на мові C#. Щоб проект міг обробляти елементи коду одночасно на HTML+CSS та C# використовується вбудований движок уявлення. На ділі виходить, що, викликаючи метод View в контролері для відображення сторінки не здійснюється її уявлення і код HTML не генерується. Контролер займається підготовкою даних і вибирає яке саме уявлення необхідно повернути в якості об'єкта ViewResult. У свою чергу ViewResult робить звернення до движку уявлення і формує візуальну частину сторінки в вихідну відповідь.

Для створення адаптивної інтерактивної системи використовувалася сітка фреймворку Bootstrap [12] [13] та набір стилів власної розробки. Це набір інструментів для створення сайтів і веб-додатків, який

поширюється у вільному доступі. Бібліотека підключається до проекту ASP.NET Core, після чого спеціалізовані класи Bootstrap можна задіяти в верстці HTML для відображення сторінок.

Особливості роботи бази даних

Оскільки бази даних для сайтів використовуються з метою зберігання різної інформації і, спрощено, представляють собою деякий набір взаємозв'язаних таблиць, необхідно вибирати інструменти для їх проектування, які допоможуть сайту працювати максимально швидко [14]. Розміри таблиць в БД різні, їх кількість довільно. Саме в базі даних зберігається на сервері необхідна для роботи інтерактивної системи інформація, наприклад, інформація про заклади, список міст України, меню ресторанів, статистичні дані і т. д.

Для інтерактивної системи пошуку закладів громадського харчування використовувався інструмент міграцій, який значно спрощує взаємодію проекту з базою даних і прискорює побудова таблиць. Таким чином, запит йде безпосередньо до всієї бази і постійно змінюється при появі в інтерактивній системі нової моделі або контролера. Завдяки такому підходу вдалося прискорити працездатність сайту, що в даній ситуації є дуже важливою складовою, адже зберігати в собі система повинна дуже багато інформації про заклади.

Після додавання контексту даних у вигляді сервісу можна надалі отримувати його в конструкторі контролера через механізм впровадження залежностей. Визначивши всі налаштування можна використовувати міграцію для створення бази даних.

Міграція [15] баз даних використовується в інтерактивній системі пошуку закладів громадського харчування, коли може виникнути ситуація, де модель змінюється. Протягом розробки проекту кілька разів доводилося додавати нові властивості. При цьому вже є база даних, де зберігаються певні дані. Щоб оновити її без видалення і створення нових таблиць використовується механізм ASP.NET MVC міграцій.

Міграції дають можливість просто додавати, по суті нові властивості при необхідності. Оскільки робота з базою даних йде через контекст даних. Це значно спрощує процесс розробки, навідміну від бесплатних CMS, де необхідно постійно оновлювати вміст бази данних за допомогою різних розширень. Також використання міграцій у проекті допомгло позбудится кешування сторінки, яке навіть після оновлення не показує зміни на сайті або робить певний функціонал не працюючим для конкретного користувача в певний момент використання системи.

В контексті необхідно додавати нові властивості в ситуації, коли в проекті з'являються нові моделі. Коли змінюється контекст даних необхідно привести міграцію від старої схеми бази даних до нової. Для цього достатньо створити нову міграцію. За допомогою конфігурацій можна створювати нові таблиці в базі даних і поля для них або вже існуючих.

Для взаємодії з базою даних в інтерактивній системі використовується розроблених контекст даних, який необхідний для підключення до бази даних через Entity Фрамеворк. Контекст даних являє собою клас, похідний від класу Двконтекст і є посередником між базою даних і висновком існуючих сутностей в проекті. Контекст даних містить одне або кілька властивостей типу DbSet <T>, де T представляє тип об'єкта, що зберігається в базі даних.

Працюючи з базовим класом контексту в нього можна додавати будь-які перерахування моделей і контролерів. Надалі це допоможе значно простіше оновлювати інтерактивну систему при необхідності. Сам процес розробки помітно прискорився з використанням такого рішення. Найголовніше вчасно здійснювати міграцію, щоб поновлення в проекті вступили в силу. Міграція проводиться після зміни контексту наступними командами: Add-Migration Initial. Після цієї команди всі зміни будуть додані за допомогою міграції в базу даних проекти. Щоб вони вступили в силу доведеться ще й оновити базу даних інтерактивної системи Update-Database.

Функціональність спроектованої системи

Використані інструменти при проектуванні інтерактивної системи допомогли мінімізувати навантаження на програмні ресурси в процесі пошуку. Додавши до цього спрощену процедуру пошуку закладів за критеріями, вдалося надати можливість користувачам менш, ніж за хвилину знайти відповідні заклади. Процес знаходження місця для обіду

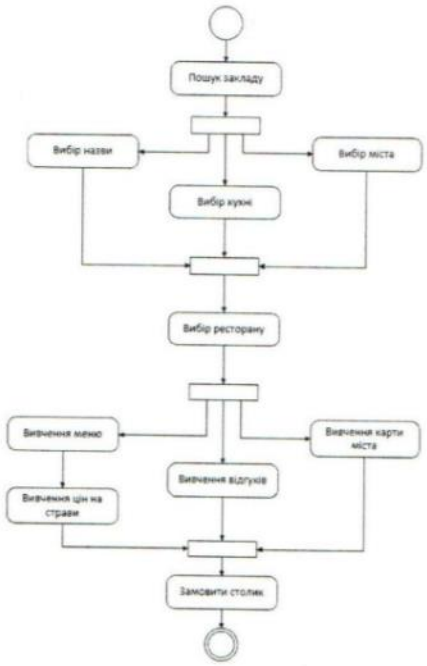


Рис. 2. Діаграма діяльності

продемонстрований на діаграмі діяльності, разом з деякими основними можливостями. Потрапляючи на головну сторінку інтерактивної системи користувач відразу ж бачить пошукову суворо, де можна знайти заклад за назвою, місту, де розташоване або кухні, яку представляє. Діаграма на Рис. 2 наочно показує, що після вибору критерієм і натискання на кнопку

"Пошук" користувач відразу ж потрапляє до "Вибору ресторану". Таким чином всього одного кліка при встановлених умовах досить, щоб побачити список з результатами. Якщо ж критерії не встановлені, то виведуться всі існуючі заклади по рейтингу на сайті.

Вибравши зі списку цікавить вас заклад здійснюється перехід на його сторінку. Саме тут можна вивчити опис, меню разом з цінами, карту розташування та відгуки від попередніх відвідувачів. Вся ця інформація плавно підводить користувача до кінцевої мети відвідування інтерактивної системи, а саме бронювання столу.

Завдяки діаграмі послідовності [16] можна побачити взаємодії між інтерактивною системою і всіма користувача, що виявиться показником її зручності. В цілому Діаграма послідовності – це спосіб Опису поведінкі системи на Основі вказівки послідовності переданих повідомлень.

На рис. 3 показаний запис протоколу конкретного сеансу роботи системи. На діаграмі послідовностівікористовується один основний тип сутности – екземпляр взаємодіючіх класіфікаторів (Користувач, Інтерактивна система, Ресторан), и один тип відношень зв'язків, за якими відбувається обмін повідомленнями. Існує кілька видів повідомлень, які в графічній нотації розрізняються видом стрілки, відповідної відношенню.

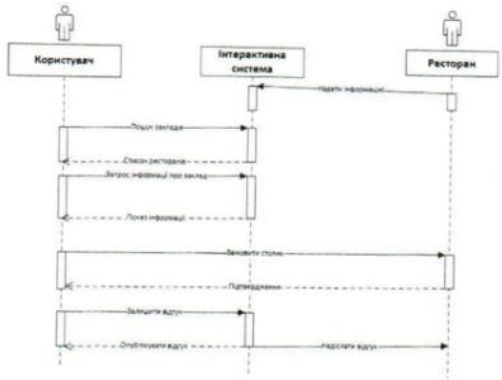


Рис. 3. Діаграма послідовності

Можна побачити, що спочатку інформація в інтерактивну систему потрапляє за допомогою власники закладів, який зарагестріровался на сайті. Також дана можливість є і у адміністратора інтерактивної системи, який через спеціальну панель управляє всією інформацією. При наявності закладів всередині інтерактивної системи користувач може за все в один клік почати пошук цікавить його місця за певними критеріями, а саме місто і кухня. Запит відправляє на обробку інтерактивної системи і у вигляді відповіді сайт видає користувачеві список всіх знайдених закладів в зручній формі. Перейшовши на сторінку конкретного закладу можна побачити детальну інформацію з фотографіями наданими власником закладу. Також на сторінці одного закладу є багато розроблених модулів, які несуть в собі дані про різні особливості конкретного місця. До таких можна віднести карту розташування, список додаткових послуг, меню з цінами, графік роботи і багато іншого.

На окрему увагу заслуговує блок меню з цінами. Ця функція не зустрічається у схожих сайтів, які вже працюють в інтернеті. Завдяки такому доповненню відвідувач інтерактивної системи може побачити, які страви пропонує той чи інший заклад. При цьому є можливість ознайомитися з основними меню, дитячим або якимось спеціальним, якщо такі присутні. Ціни на страви вказані саме ті, які діють в закладі на поточний момент. При цьому в зручній формі виводиться середня вартість страви, що дає можливість ще до відвідування місця прикинути у скільки вам обійдеться обід або вечерю.

Функція бронювання столика є унікальним відмінністю інтерактивної системи пошуку закладів громадського харчування від конкурентів. Після перехід на сторінку цікавить користувача закладу в правій частині екрану, він побачить форму для заповнення. Тут знадобиться вказати ім'я, адресу електронної пошти, номер телефону, дату і час броні, а також кількість осіб, що доведеться в заклад. Заповнена форма відправляється одним натисканням і відправляється відразу ж на пошту власників закладу. Вже там її обробляє адміністратор ресторану або менеджер. Залежно від швидкості роботи персоналу час очікування відповіді може відрізнятися. Проте взаємодія з адміністраторами закладу безпосередньо робить бронювання столика максимально простим і зручним. Після відправки заявки на мобільний телефон і одночасно на пошту прийде повідомленні про успішне бронювання. Якщо ж місць в закладі немає, то з вами зв'яжуться по телефону, щоб повідомити про це або запропонувати інші варіанти. На відміну від сайтів конкурентів в інтерактивній системі столик бронюється прямо зі сторінки закладу, що не можуть надати інші сайти. Як правило, на них відсутня така фнукцій, якщо ж звернути увагу на сервіс Google Maps, який один з найбільш затребуваних в наш час для пошуку різних закладів, то тут доведеться перейти на сайт ресторану, кафе або бару і бронювати столик вже через нього, що займає надто багато часу.

Можливість залишати відгуки була спеціально реалізована для того, щоб власники ресторанів, кафе і барів розуміли, що люди думають про них. Після того, як відгук з'являється на сторінці з закладом його власник може легко прочитати повідомлення, проаналізувати інформацію отриману від гостей і зробити певні висновки. Інтерактивна система пошуку закладів громадського харчування виступає ще і в ролі зручного інструменту комунікації між відвідувачами і власниками закладів.

Система, яка була розроблена в результаті має наступні можливості:

– В інтерактивній системі можна розмістити заклад, заповнити його опис, додати фотографії, створити меню з розцінками, вказати адресу, особливості та графік роботи.

– Реалізована зручна система пошуку за такими критеріями, як назва, місто і кухня. Пошук здійснюється в один клік.

– Користувач може забронювати столик на сторінці з закладом заповнивши форму з такими критеріями, як ім'я, адреса електронної пошти, номер телефону, дата бронювання і кількість осіб.

– Розроблено додатковий функціонал для меню закладу в зручній формі для користувачів. Детальний меню дає можливість ознайомитися з цінами і виводить середню вартість чека в цьому закладі.

– Відвідувач може побачити місце розташування закладу на карті для зручності вибору маршруту до нього.

– Кожен може ділитися своїми враженнями про заклад. Реалізована система відгуків з виставленням оцінки. В результаті виводиться середня оцінка закладу, що потім допомагає групувати їх в загальному каталозі якщо користувач вибирає критерій сортівки "За рейтингом".

– Зручна адміністратівная панель дозволяє швидко заповнити всі блоки з описом для конкретного місця.

– Додаткова функція блогів дозволяє зареєстрованим користувачам писати статті на тему ресторанної справи, описувати власні рецепти і створювати повноцінні огляди на заклади.

Таким чином результатом розробки стала інтерактивна система пошуку закладів громадського харчування [17] [18], яка представляє собою сайт-каталог зі зручним інтерфейсом. В системі були реалізовані усі заплановані можливості та завдяки адаптивності стало можливим використання на комп'ютерах, планшетах і смартфонах під управлінням різніх операційних систем.

Висновки

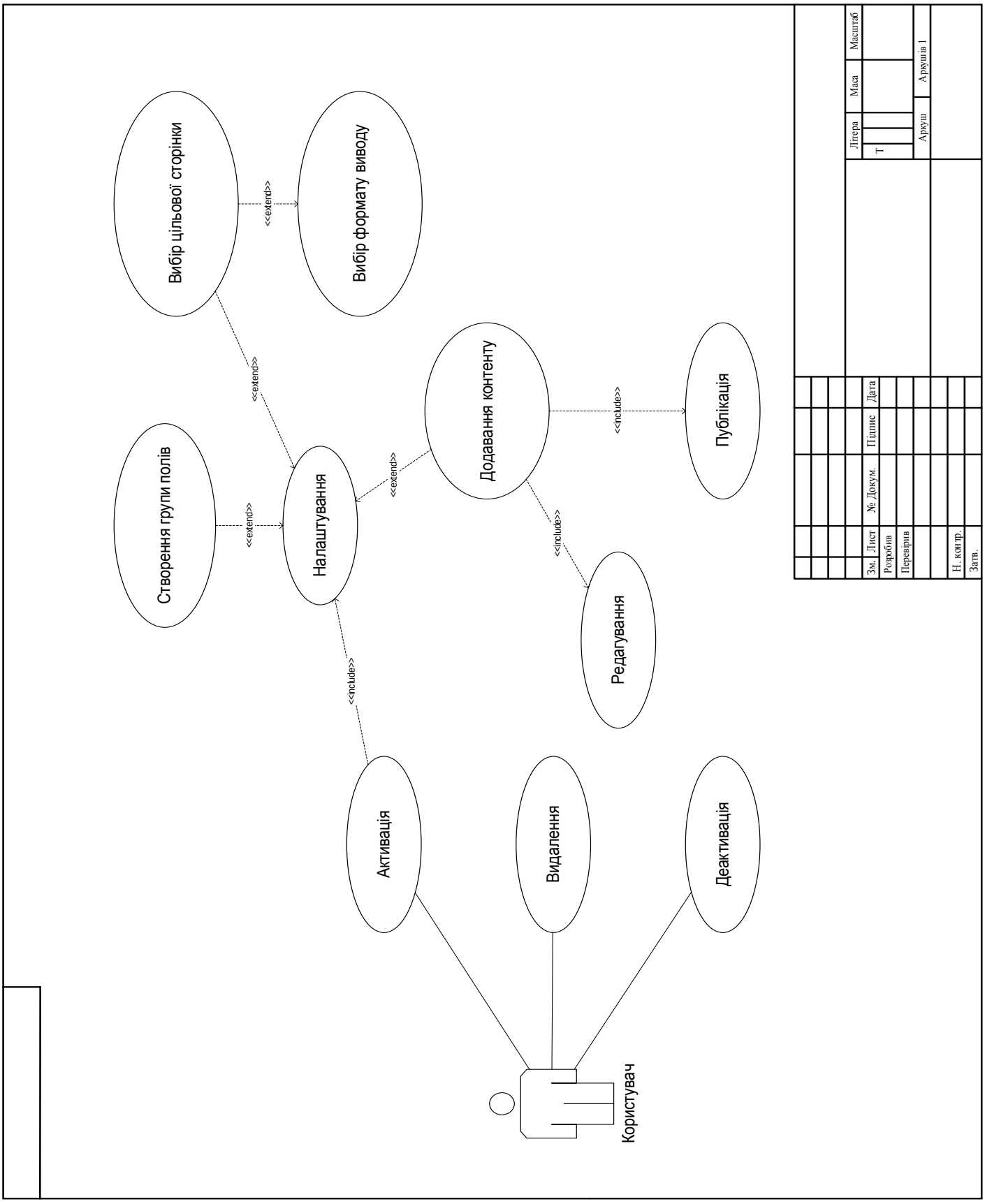
Використання вище перерахованих іструментів дозволило створити інтерактивну систему з великим каталогом закладів громадського харчування: ресторанів, кафе, барів, їдалень та інших. Зручна система пошуку дозволяє знаходити заклад за різними критеріями: місту, де розташовується, кухні та назві [19]. Розробивши певні інструменти для існуючого патерна MVC вдалося значно простіше реалізувати функціонал системи та отримати можливість комфортно редагувати або тестувати проект.

Перш за все система стане корисна туристам, які будуть мати бажання відвідати українські заклади громадського харчування. Для жителів нашої країни це також зручний інструент пошуку достойного закладу. З точки зору власника ресторану або цілої мережі використання інтерактивної системи допоможе стати більш популярним. Надалі інтерактивна система може поповнитися новими закладами за межами України, рекламними можливостями і різними інструментами для більш докладного і зручного опису кожного закладу.

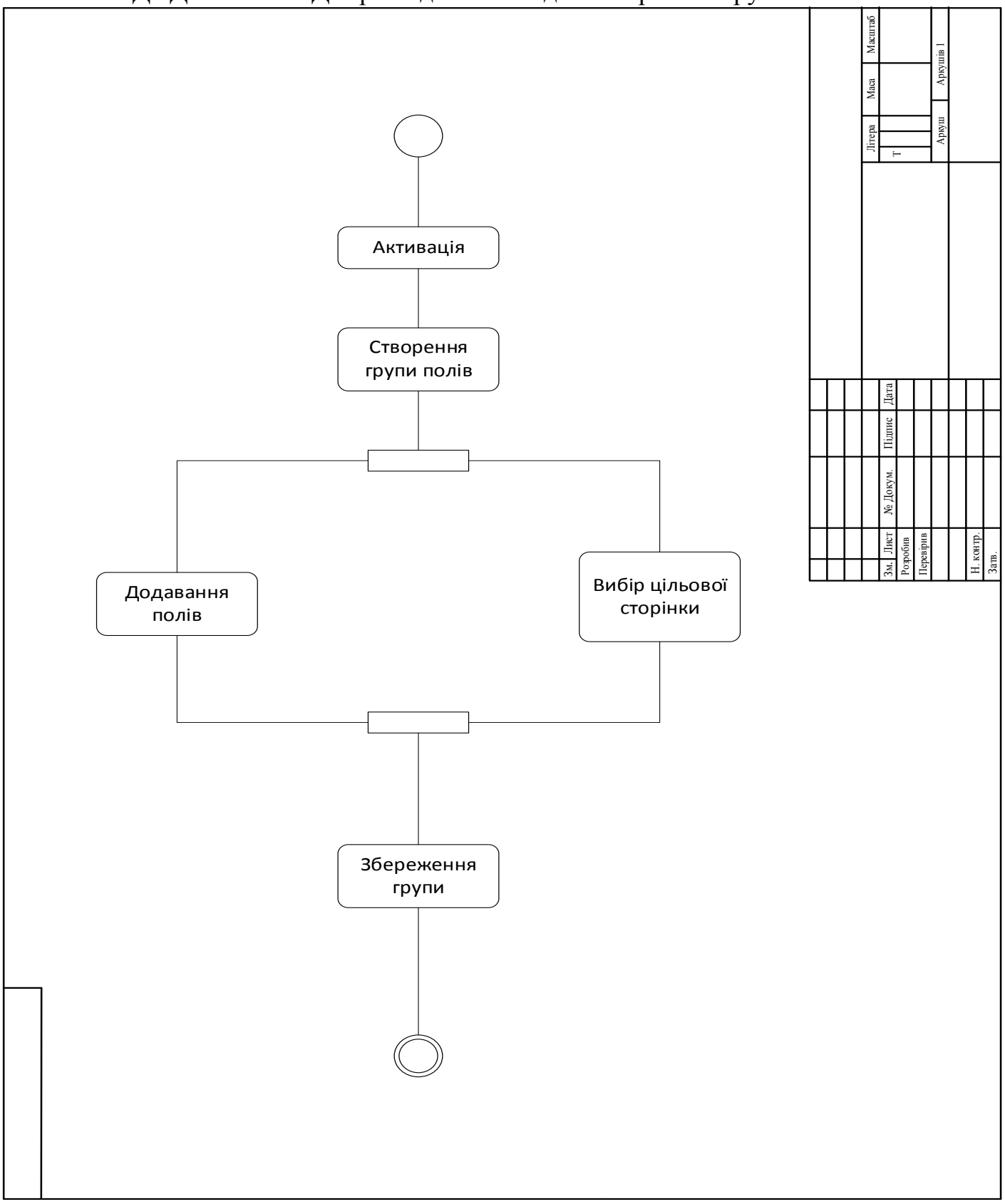
Література

1. Сервіс пошуку закладів громадського харчування по карті – Режим доступу: https://www.google.ru/maps/search
2. ASP.NET Core MVC з прикладами на C # для професіоналів 2017 г. - Фрімен А.
3. ASP.NET MVC 5 на C # 5.0 для професіоналів 2015 г. - Фрімен А.
4. Движок уявлень і Razor - Режим доступу: https://metanit.com/sharp/mvc/4.3.php
5. Мова програмування C # 5.0 і платформа .NET 4.5, 2015 г. - Троелсен Е.
6. Прийоми об'єктно-орієнтованого проектування. Патерни проектування, 2014 г. - Гамма, Хелм, Джонсон, Вліссідес
7. JavaScript. Детальний керівництво, 2008 - Фленаган Д.
8. CSS рецепти програмування, 2015 г. - Крістофер Шмітт
9. Введення в системи баз даних, 2003 г. - К. Дж. Дейт
10. jQuery. Докладне керівництво по просунутому JavaScript, 2008 - Бібо Б., Кац І.
11. Ajax для новачків - Режим доступу: https://habrahabr.ru/post/14246/
12. Робота з Bootstrap – Режим доступу: http://mybootstrap.ru/get-started/
13. Чуйний веб-дизайн, 2011 р - Ітан Маркотт
14. Доступ до данних в ASP.NET Core – Режим доступу: https://msdn.microsoft.com/ ru-ru/magazine/ mt742867.aspx
15. Миграції. Режим доступу: http://laravel.su/ docs/5.2/migrations
16. Мова UML. Керівництво користувача (2-е изд.), 2006 р - Буч Г., Рамбо Д., Якобсон І.
17. Досконалий код. Практичний посібник з розробки програмного забезпечення, 2005 р - Макконнелл С.
18. Entity Framework Code First на практиці – Режим доступу: https://habrahabr.ru /post/236037/
19. Фільтри пошукової системи Google – Режим доступу: https://seoprofy.ua/blog/poiskovye-sistemy/ google-filters
20. Методичні вказівки до виконання курсових проектів з дисципліни «Комп'ютерна електроніка» для студентів напряму підготовки 6.050201 «Системна інженерія» кафедри Автоматики та управління у технічних системах / Укл.: А.О. Новацький – К: НТУУ „КПІ", 2013 - 106 с.
21. Оформлення текстових документів у навчальному процесі. Стандарт організації (кафедри) СОУ АУТС 01-15. Для студентів кафедри автоматики та управління в технічних системах.

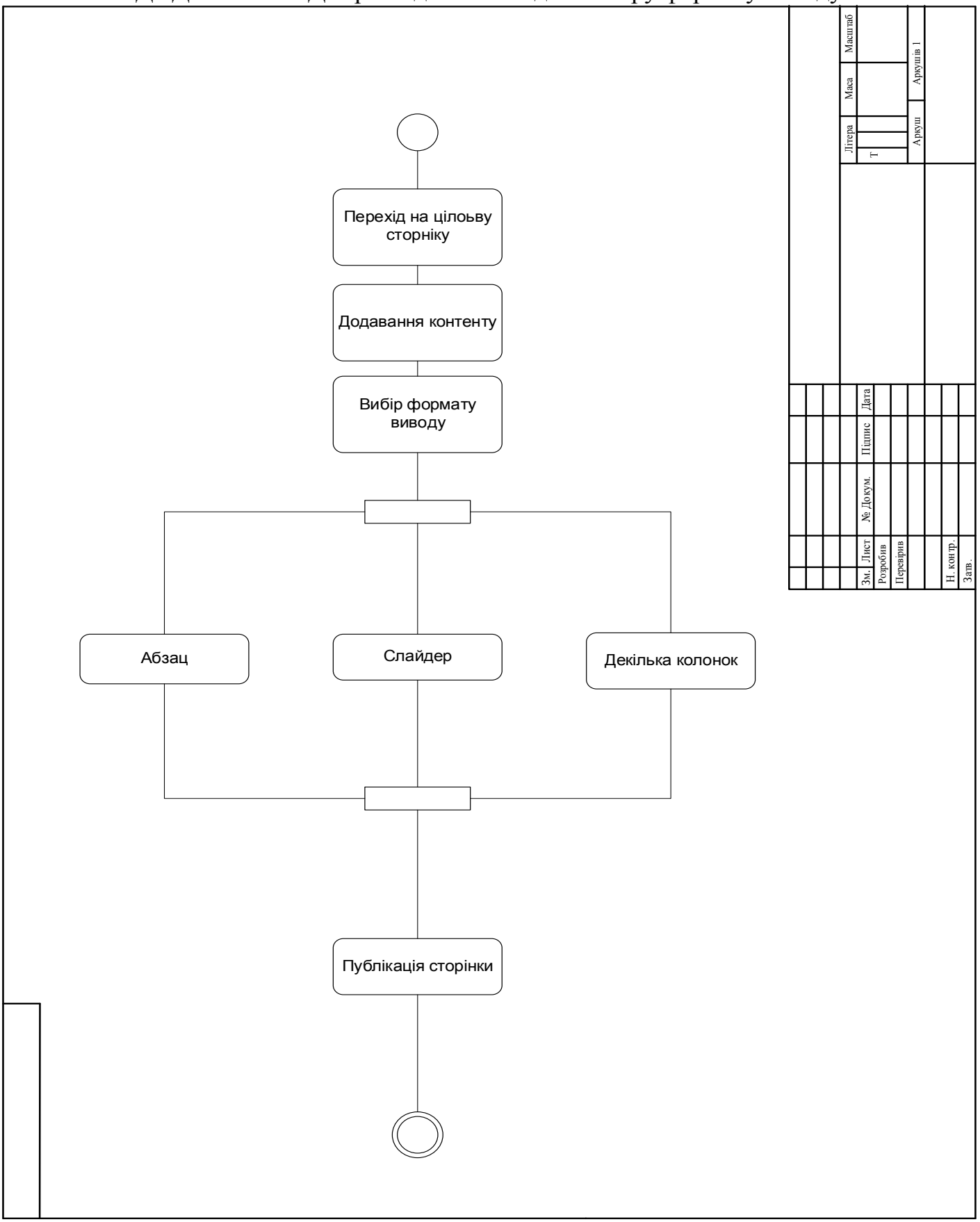
ДОДАТОК Б – Діаграма використання



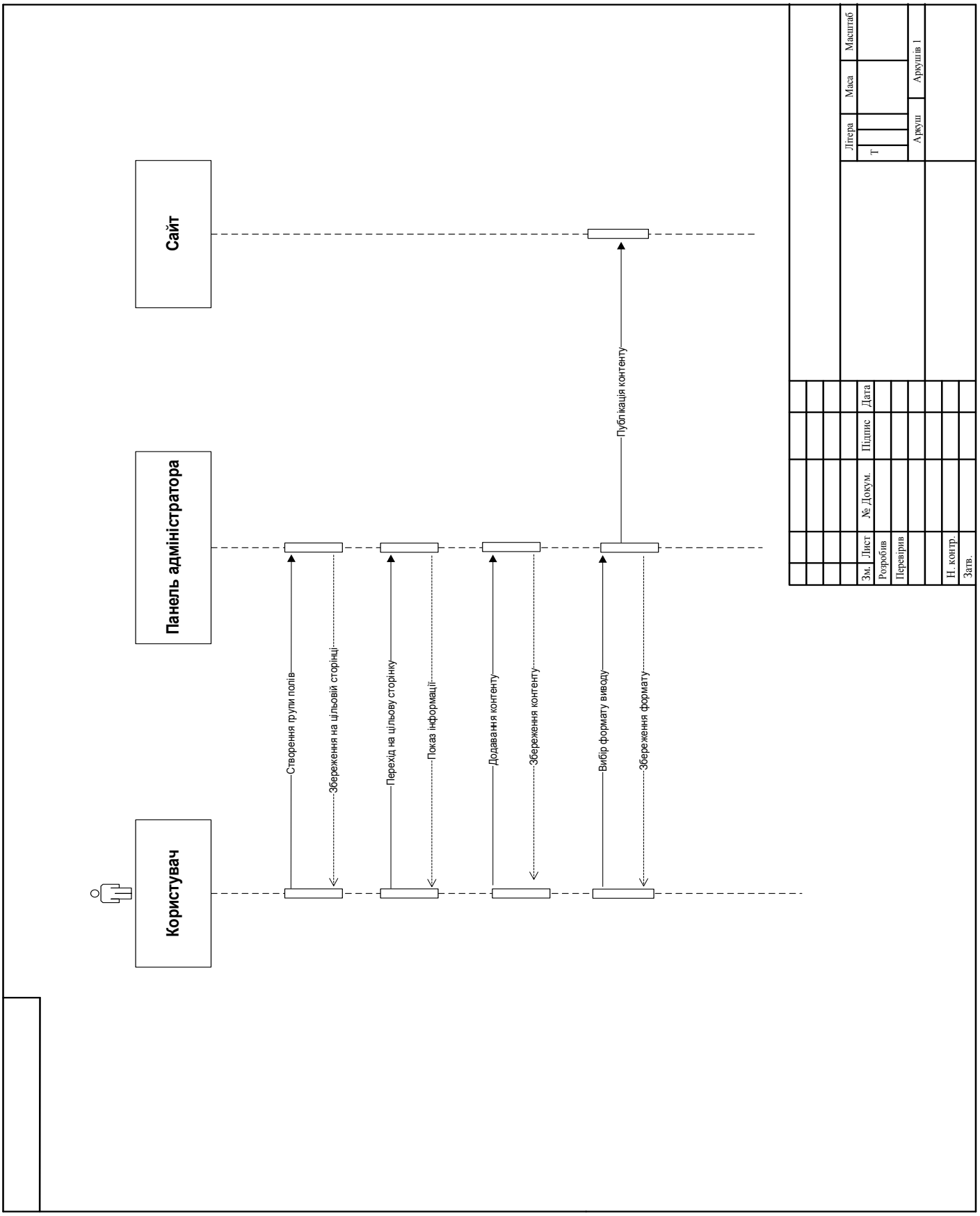
ДОДАТОК В – Діаграма діяльності для створенная группи полів



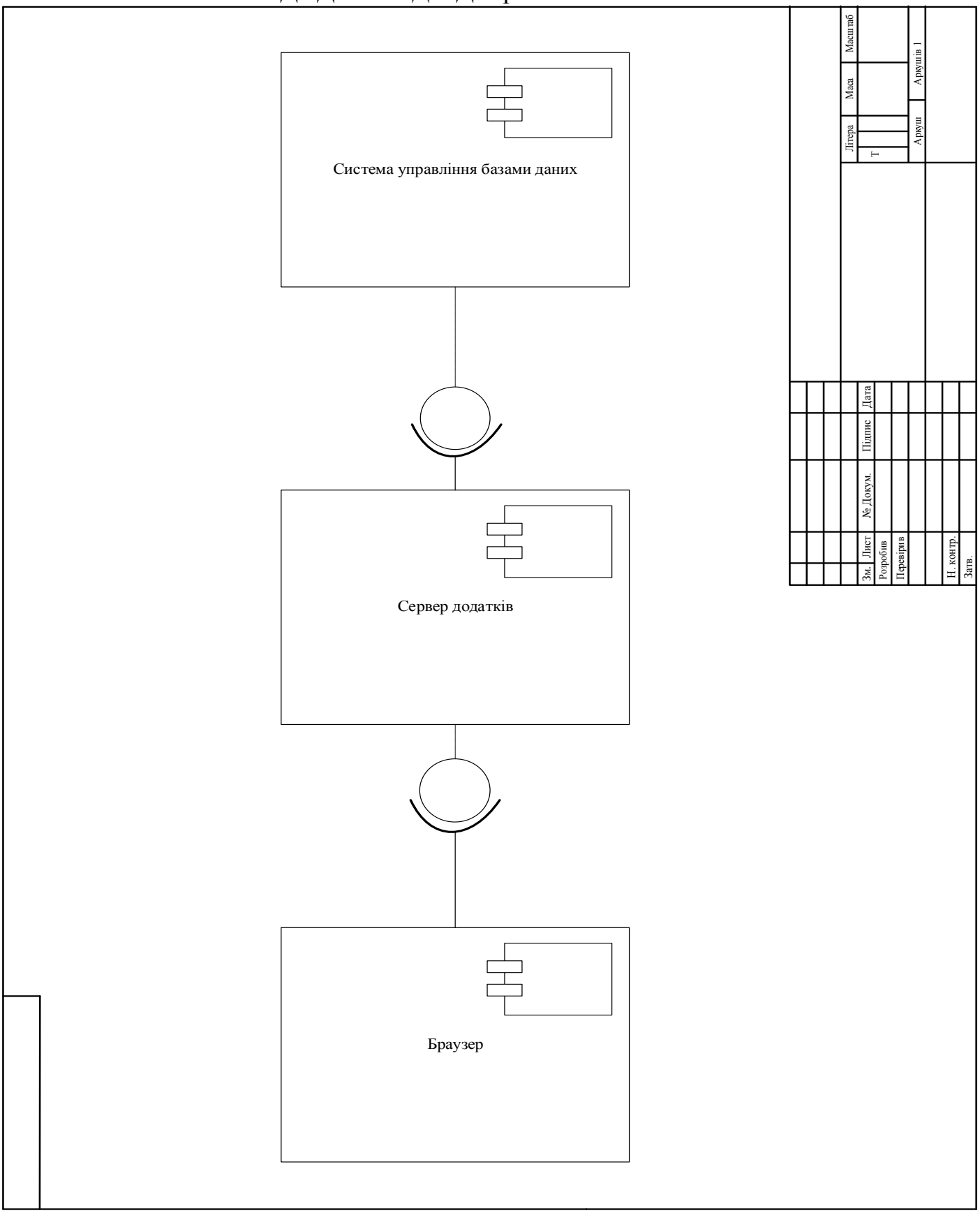
ДОДАТОК Г – Діаграма діяльності для вибору формату виводу



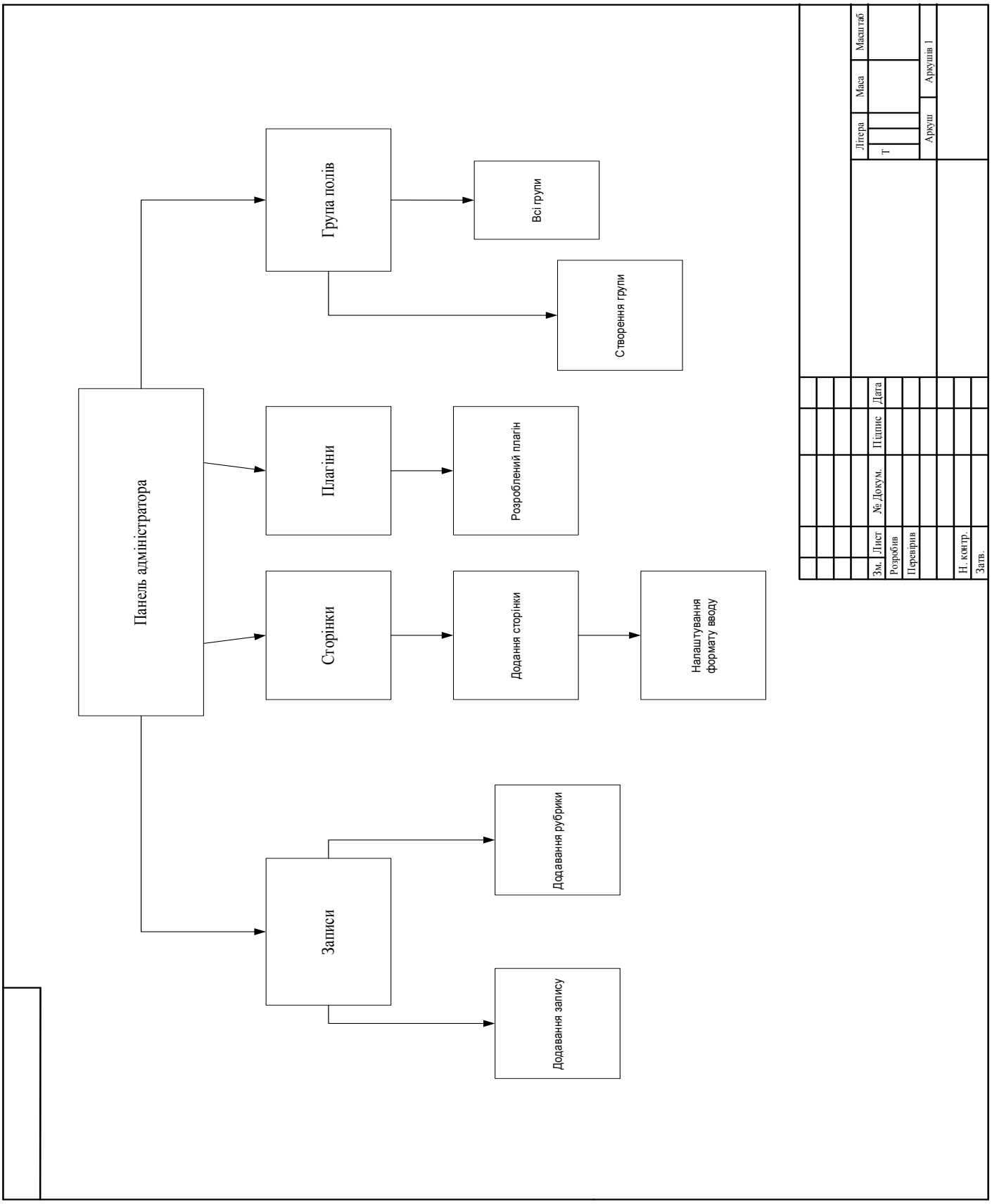
ДОДАТОК Ґ – Діаграма послідовності



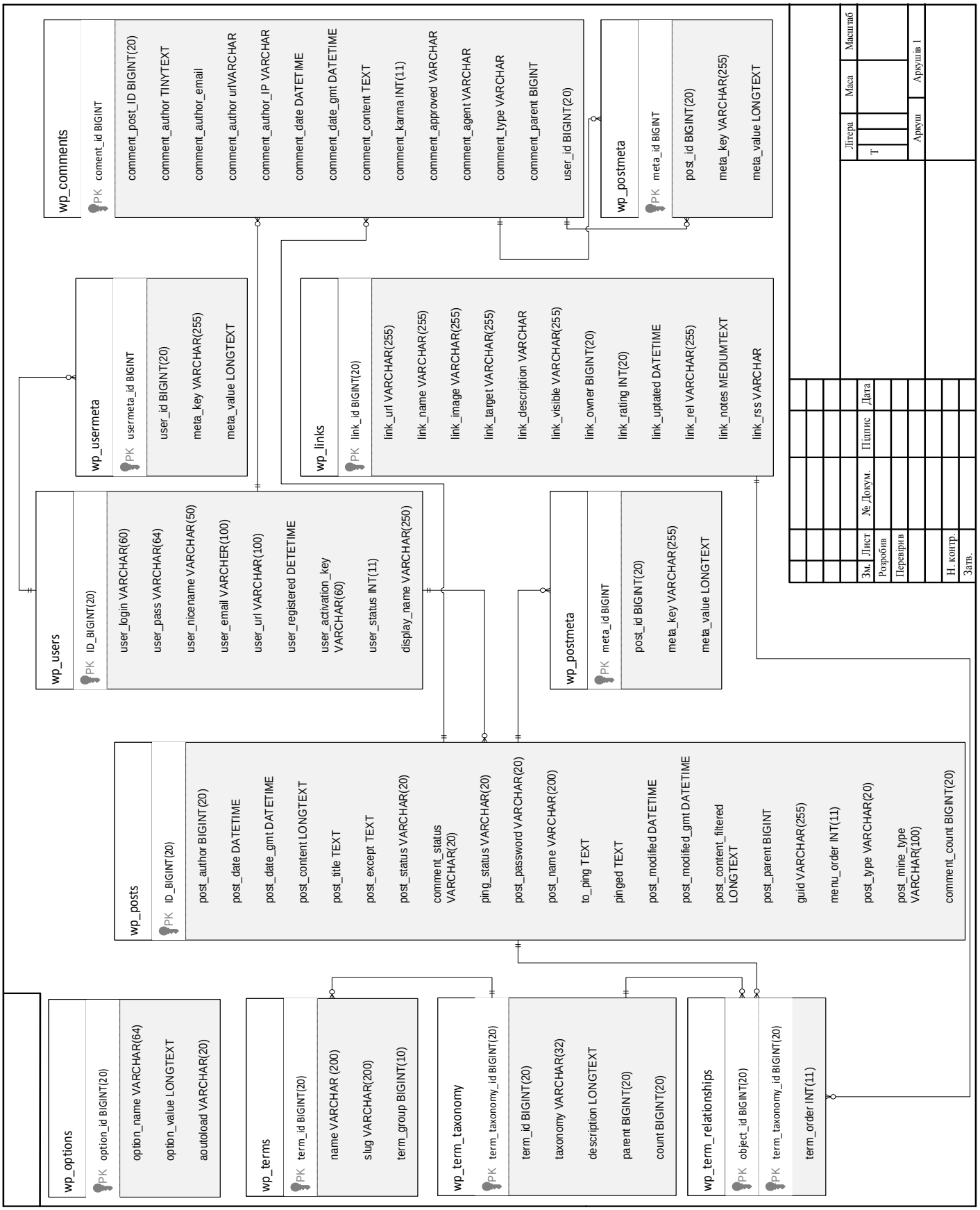
ДОДАТОК Д – Діаграма компонентів



ДОДАТОК Е – Структурна схема плагін



ДОДАТОК Є – Схема бази даних для блогу



ДОДАТОК Ж – Діаграмма класів